



Proceedings of the Eighth International Conference on
Computational Creativity

ICCC 2017
Atlanta — 19 - 23 June

Ashok Goel, Anna Jordanous, Alison Pease (Editors)

Supported by the Association for Computational Creativity
(ACC)



Georgia Institute of Technology
Atlanta, Georgia, US

<http://computationalcreativity.net/iccc2017/>

First published 2017

TITLE: Proceedings of the 8th International Conference on Computational Creativity
(ICCC'17)

EDITORS: Ashok Goel, Anna Jordanous, Alison Pease

ISBN: 978-0-692-89564-1

Supported by the Association for Computational Creativity (ACC)

ISBN 978-0-692-89564-1



Program Committee

Senior PC Members

Oliver Bown - University of New South Wales
F. Amílcar Cardoso - University of Coimbra
Simon Colton - Falmouth University / Goldsmiths College, University of London
Arne Eigenfeldt - Simon Fraser University
Pablo Gervás - Universidad Complutense de Madrid
Kazjon Grace - University of North Carolina at Charlotte
Nada Lavrač - Jozef Stefan Institute
Mary Lou Maher - University of North Carolina Charlotte
Ruli Manurung - Faculty of Computer Science, Universitas Indonesia
Rafael Pérez y Pérez - Universidad Autónoma Metropolitana at Cuajimalpa
Rob Saunders - Falmouth University
Hannu Toivonen - University of Helsinki
Dan Ventura - Brigham Young University
Geraint Wiggins - Queen Mary, University of London

PC Members

Margareta Ackerman - San Jose State University
Kat Agres - Agency for Science, Technology and Research (A*STAR)
Wendy Aguilar - IIMAS, UNAM
Tarek Richard Besold - University of Bremen
Debarun Bhattacharjya - IBM T.J. Watson Research Center
Josep Blat - Universitat Pompeu Fabra
Daniel Brown - Cheriton School of Computer Science, University of Waterloo
David C. Brown - CS Dept., WPI
Andrew Brown - Queensland Conservatorium, Griffith University
Michael Cook - Goldsmiths, University of London
João Correia - CISUC-DEI
Joe Corneli - University of Edinburgh
Vincent Corruble - LIP6, Université Pierre et Marie Curie (Paris 6)
Mark d'Inverno - Goldsmiths, University of London
Jim Davies - Carleton University
Nicholas Davis - Georgia Institute of Technology
Sarah Fdili Alaoui - LRI-Université Paris-Sud 11
Katherine Fu - Georgia Institute of Technology
Luís Góes - Pontifícia Universidade Católica de Minas Gerais
Andrés Gómez de Silva Garza - Instituto Tecnológico Autónomo de México
Hugo Gonçalo Oliveira - CISUC, University of Coimbra
Jeremy Gow - Goldsmiths, University of London
Sascha Griffiths - Universität Hamburg
Christian Guckelsberger - Goldsmiths, University of London
Ivan Guerrero - UNAM-IIMAS
Matthew Guzdial - Georgia Institute of Technology
Sarah Harmon - University of California Santa Cruz
Raquel Hervás - Universidad Complutense de Madrid

Amy K. Hoover - University of Central Florida
Mikhail Jacob - Georgia Institute of Technology
Colin Johnson - University of Kent
Maximos Kaliakatsos-Papakostas - Aristotle University of Thessaloniki
Anna Kantosalo - University of Helsinki
Bill Keller - University of Sussex
Oliver Kutz - Free University of Bozen-Bolzano
Carolyn Lamb - University of Waterloo
Carlos León - Universidad Complutense de Madrid
Antonios Liapis - Institute of Digital Games, University of Malta
Heather Ligler - Georgia Institute of Technology
Simo Linkola - University of Helsinki
Julie Linsey - Georgia Institute of Technology
Maria Teresa Llano Rodriguez - Goldsmiths, University of London
Phil Lopes - Institute of Digital Games, University of Malta
Róisín Loughran - University College Dublin
Todd Lubart - University Paris Descartes
Brian Magerko - Georgia Institute of Technology
Thor Magnusson - Music Department, University of Sussex
Jon McCormack - Monash University
Stephen McGregor - Queen Mary University of London
David Meredith - Aalborg University
Diarmuid O'Donoghue - National University of Ireland, Maynooth
Ana-Maria Olteteanu - SFB-TR8 Spatial Cognition, Bremen Universitt
Philippe Pasquier - Simon Fraser University
Alexandre Miguel Pinto - University Of Coimbra
Enric Plaza - IIIA-CSIC
Senja Pollak - Jozef Stefan Institute
Marco Schorlemmer - IIIA-CSIC
Marco Scirea - IT University of Copenhagen
Emily Short - Freelance
Mei Si - Rensselaer Polytechnic Institute (RPI)
Adam M. Smith - University of California Santa Cruz
Gillian Smith - Northeastern University
Ricardo Sosa - Auckland University of Technology
Tristan Strange - University of Kent
Bob Sturm - Queen Mary University of London
Anne Sullivan - University of Central Florida
Tapio Takala - Aalto University
Julian Togelius - New York University
Tatsuo Unemi - Soka University
Frank van der Velde - University of Twente
Lav Varshney - University of Illinois at Urbana-Champaign
Swaroop Vattam - MIT Lincoln Lab
Anna Weisling - Georgia Institute of Technology
Georgios N. Yannakakis - Institute of Digital Games, University of Malta
Matthew Yee-King - Goldsmiths, University of London
Jichen Zhu - Drexel University

Preface

This volume contains the papers presented at ICCC 2017, the 8th International Conference on Computational Creativity held in Atlanta, Georgia, USA from June 19th - June 23rd, 2017 <http://computationalcreativity.net/iccc2017/>. The conference was hosted at the Georgia Institute of Technology.

Computational creativity is the art, science, philosophy and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative. As a field of research, this area is thriving, with progress in formalising what it means for software to be creative, along with many exciting and valuable applications of creative software in the sciences, the arts, literature, gaming and elsewhere. The ICCC conference series, organized by the Association for Computational Creativity since 2010, is the only scientific conference that focuses on computational creativity alone and also covers all its aspects.

We received 69 paper submissions, in five categories:

1. Technical papers advancing the state of art in research [papers posing and addressing hypotheses about aspects of creative behaviour in computational systems];
2. System and resource description papers [papers describing the building and deployment of a creative system or resource to produce artefacts of potential cultural value in one or more domains];
3. Study papers presenting enlightening novel perspectives [papers which draw on allied fields such as psychology, philosophy, cognitive science, mathematics, humanities, the arts, and so on; or which appeal to broader areas of Artificial Intelligence and Computer Science in general; or which appeal to studies of the field of Computational Creativity as a whole];
4. Cultural application papers [papers presenting the usage of creative software in a cultural setting];
5. Position papers arguing for an opinion [papers presenting an opinion on some aspect of the culture of Computational Creativity research, including discussions of future directions, past triumphs or mistakes and issues of the day].

Each submission was reviewed by 3 program committee members and then discussed among the reviewers, if needed, to resolve controversial and borderline cases. Senior Program Committee Members led discussions and also prepared recommendations based on the reviews and discussions. In total, around 300 reviews and meta-reviews were carried out in the process. Papers were accepted based on quality, academic rigour and relevance to one or more of the conference's five paper categories.

The committee accepted 34 full papers. Papers were presented either as oral presentations, posters or demos, depending on the nature of the contribution. The three-and-a-half days of the ICCC 2017 scientific program consisted in a series of exciting sessions for oral presentations of papers and a special session for posters and demos.

The program included two keynote talks. The first was by Dr. Milena Fisher, Co-Founder and President of The Creativity Post, and the second was by Professor Gil Weinberg, from the School of Music Technology at Georgia Tech.

This conference included a record number of satellite events related to creativity and computers, including three workshops, two tutorials and a Doctoral Consortium. The three workshops were the 5th International Workshop on Musical Metacreation (MUME); a joint

workshop between the Co-Creation Workshop and Working on Computational Creativity and Games (CCGW); and Computational Creativity and Social Justice (CCSJ). The MUME workshop also hosted a concert of musical metacreation. The tutorials organised were Literary Creativity and Narrative Generation and Tweet Dreams Are Made Of This: Building Creative Twitterbots. We also held two panel sessions: Computational Creativity and Design; and Computational Creativity and Discovery.

ICCC 2017 gave several awards including the Best Paper Award and the Best Student Paper Award.

We thank our sponsors, from which we received very useful support: US National Science Foundation, Artificial Intelligence, Georgia Institute of Technology, and the Georgia Tech Gvu Center. We thank the program committee and the senior program committee for their hard work in reviewing papers and the EasyChair platform that made our work easier. We also thank all those involved in organising ICCC 2017, the ACC steering committee, best paper reviewers and those involved in organising and supporting the workshops, tutorials and doctoral consortium.

ICCC 2017 organising committee

GENERAL CHAIR: Ashok Goel, Georgia Institute of Technology

PROGRAM CO-CHAIR: Anna Jordanous, University of Kent

PROGRAM CO-CHAIR: Alison Pease, University of Dundee

WORKSHOP CO-CHAIR: Kazjon Grace, University of North Carolina

WORKSHOP CO-CHAIR: Ruli Manurung, Google

LOCAL CHAIR: Mikhail Jacob, Georgia Institute of Technology

LOCAL COMMITTEE: Jeff Collins, Georgia Institute of Technology

Heather Ligler, Georgia Institute of Technology

Gerard Roma, Georgia Institute of Technology

Anna Xambó, Georgia Institute of Technology

MEDIA CHAIR: Matthew Guzdial, Georgia Institute of Technology

MEDIA COMMITTEE: Anna Weisling, Georgia Institute of Technology

June 2017

Table of Contents

<i>Characteristics of Pro-c Analogies and Blends between Research Publications</i> ... 1 Yalemisew Abgaz, Diarmuid O'Donoghue, Donny Hurley, Ehtzaz Chaudhry and Jian Jun Zhang	
<i>Teaching Computational Creativity</i> 9 Margareta Ackerman, Ashok Goel, Colin Johnson, Anna Jordanous, Carlos León, Rafael Pérez y Pérez, Hannu Toivonen and Dan Ventura	
<i>Early-creative behavior: the first manifestations of creativity in a developmental agent</i> 17 Wendy Aguilar and Rafael Pérez y Pérez	
<i>Expanding and Weighting Stereotypical Properties of Human Characters for Linguistic Creativity</i> 25 Khalid Alnajjar, Mika Hämäläinen, Hanyang Chen and Hannu Toivonen	
<i>Creative Flavor Pairing: Using RDC Metric to Generate and Assess Ingredients Combinations</i> 33 Álvaro Amorin, Luís Fabrício Góes, Alysson Silva and Celso França	
<i>Creative Robot Dance with Variational Encoder</i> 41 Agnese Augello, Emanuele Cipolla, Ignazio Infantino, Giovanni Pilato, Adriano Manfrè and Filippo Vella	
<i>Text Transformation Via Constraints and Word Embedding</i> 49 Benjamin Bay, Paul Bodily and Dan Ventura	
<i>Human-Level Concept Learning in Computational Creativity</i> 57 Paul Bodily, Benjamin Bay and Dan Ventura	
<i>How Can We Deal With The Design Principle Of Visibility In Highly Encapsulated Computationally Creative Systems</i> 65 Liam Bray, Oliver Bown and Benjamin Carey	
<i>A Unit Selection Methodology for Music Generation using Deep Neural Networks</i> 72 Mason Bretan, Gil Weinberg and Larry Heck	
<i>A Pig, an Angel and a Cactus Walk Into a Blender: A Descriptive Approach to Visual Blending</i> 80 João Cunha, João Gonçalves, Pedro Martins, Penousal Machado and F. Amílcar Cardoso	
<i>Distributed Musical Decision-making in an Ensemble of Musebots: Dramatic Changes and Endings</i> 88 Arne Eigenfeldt, Oliver Bown, Andrew Brown and Toby Gifford	
<i>CAN: Creative Adversarial Networks: Generating “Art” by Learning About Styles and Deviating from Style Norms</i> 96 Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny and Marian Mazzone	
<i>Human-Robot Co-Creativity: Task Transfer on a Spectrum of Similarity</i> 104 Tesca Fitzgerald, Ashok Goel and Andrea Thomaz	

<i>Blend City, BlendVille</i>	112
João Gonçalves, Pedro Martins and Amílcar Cardoso	
<i>Encouraging p-creative behaviour with computational curiosity</i>	120
Kazjon Grace, Mary Lou Maher, Maryam Mohseni and Rafael Perez Y Perez	
<i>Addressing the “Why?” in Computational Creativity: A Non-Anthropocentric, Minimal Model of Intentional Creative Agency</i>	128
Christian Guckelsberger, Christoph Salge and Simon Colton	
<i>Narrative-inspired Generation of Ambient Music</i>	136
Sarah Harmon	
<i>The Creative Machine</i>	143
James Hodson	
<i>Learning to Create Jazz Melodies Using a Product of Experts</i>	151
Daniel Johnson, Robert Keller and Nicholas Weintraut	
<i>Co-creativity and perceptions of computational agents in co-creativity</i>	159
Anna Jordanous	
<i>DeepTingle</i>	167
Ahmed Khalifa, Gabriella A. B. Barros and Julian Togelius	
<i>Fluidic Games in Cultural Contexts</i>	175
Mark J. Nelson, Swen Gaudl, Simon Colton, Edward J. Powley, Blanca Perez Ferrer, Rob Saunders, Peter Ivey and Michael Cook	
<i>Incorporating novelty, meaning, reaction and craft into computational poetry: a negative experimental result</i>	183
Carolyn Lamb, Daniel Brown and Charles Clarke	
<i>Aspects of Self-awareness: An Anatomy of Metacreative Systems</i>	189
Simo Linkola, Anna Kantosalu, Tomi Männistö and Hannu Toivonen	
<i>Application Domains Considered in Computational Creativity</i>	197
Róisín Loughran and Michael O’Neill	
<i>Stay Awhile and Listen to 3Buddy, a Co-creative Level Design Support Tool</i> ..	205
Pedro Lucas and Carlos Martinho	
<i>Applying Narrative Theory to Aid Unexpectedness in a Story Generation System</i>	213
Todd Pickering and Anna Jordanous	
<i>A Model of Inter-musician Communication for Artificial Musical Intelligence</i> ..	221
Oscar Puerto and David Thue	
<i>A Ballad of the Mexicas: Automated Lyrical Narrative Writing</i>	229
Divya Singh, Margareta Ackerman and Rafael Pérez y Pérez	
<i>Unified Classification and Generation Networks for Co-Creative Systems</i>	237
Kunwar Yashraj Singh, Nicholas Davis, Chih-Pin Hsiao, Ricardo Macias, Brenda Lin and Brian Magerko	
<i>Déjà Vu All Over Again: On the Creative Value of Familiar Elements in the Telling of Original Tales</i>	245
Tony Veale	

<i>How to Build a CC System</i>	253
Dan Ventura	
<i>Generating Animations by Sketching in Conceptual Space</i>	261
Tom White and Ian Loh	

Characteristics of Pro-c Analogies and Blends between Research Publications

¹Yalemisew Abgaz, ¹Diarmuid P. O'Donoghue, ²Donny Hurley,
³Ehtzaz Chaudhry, ³Jian Jun Zhang

¹ Department of Computer Science, Maynooth University, Maynooth, Co. Kildare, Ireland.
Diarmuid.ODonoghue@nuim.ie

² Department of Computing, Institute of Technology Sligo, Ash Lane, Sligo, Ireland.

³ National Centre for Computer Animation, Bournemouth University, Bournemouth, UK.

Abstract

Dr Inventor is a tool that aims to enhance the professional (Pro-c) creativity of researchers by suggesting novel hypotheses, arising from analogies between publications. Dr Inventor processes original research documents using a combination of lexical analysis and cognitive computation to identify novel comparisons that suggest new research hypotheses, with the objective of supporting a novel research publication. Research on analogical reasoning strongly suggests that the value of analogy-based comparisons depends primarily on the strength of the mapping (or counterpart projection) between the two analogs. An evaluation study of a number of computer generated comparisons attracted creativity ratings from a group of practicing researchers. This paper explores a variety of theoretically motivated metrics operating on different conceptual spaces, identifying some weak associations with users' creativity ratings. Surprisingly, our results show that metrics focused on the mapping appear to have less relevance to creativity than metrics assessing the inferences (blended space). This paper includes a brief description of a research project currently exploring the best research hypothesis generated during this evaluation. Finally, we explore PCA as a means of specifying a combined multiple metric to detect comparisons to enhance researchers' creativity.

Introduction

Analogical thinking was a frequent mode of thought for eminent scientists like Faraday, Maxwell and Kepler. This paper concerns an analogy-based model to enhance the creativity of practicing scientists by employing a computational model of analogy to uncover novel and potentially useful comparisons between research papers. In order to support its users, Dr Inventor (O'Donoghue, Abgaz, Hurley, & Ronzano, 2015) generates analogy-based comparisons that are similar to those developed by scientists - were they to explore the same comparisons "manually". This objective is achieved through a realistic computational simulation (Gentner & Smith, 2012), building upon several decades of focused work on analogical comparisons and conceptual blending (Fauconnier & Turner, 1998). Computational modelling of analogy has relied primarily on human constructed data (O'Donoghue & Keane, 2012), but Dr Inventor outlined in this paper uses raw data sourced directly from existing

research publications. This paper focuses on identifying the most creative comparisons for a given target, so the user only explores the most *Pro-c* creative (Kaufman & Beghetto, 2009) hypotheses.

Analogical comparisons can make a problematic concept seem more familiar, but can also make familiar ideas seem novel and fresh by comparison to some unexpected source. Novel and potentially creative comparisons can highlight previously overlooked facets of the original concept, bringing to light such information. Searching for novel analogies might be one specific mode of the divergent thinking associated with creativity. Potential applications of the Dr Inventor system presented in this paper range from creativity assistant to plagiarism detection (Hurley, Abgaz, Ali, & O'Donoghue, 2016). Thus, the main objective of this paper is to identify qualities and metrics that support the accurate identification of analogous pairs of publications that have the greatest impact on users' creativity.

This paper begins with a review of related work and some background on analogy and conceptual blending. We outline the Dr Inventor model before presenting users' evaluations for a collection of inter-publication comparisons. Several theoretically motivated metrics are statistically examined as predictors of creativity ratings. We also briefly outline a research project that arose from one of Dr Inventor's research hypotheses.

Background and Related Work

The IBM Watson (Pinel & Varshney, 2014) cognitive computing system incorporates a deep parsing of natural language documents, enabling some recent forays into culinary creativity as IBM Chef Watson. While Dr Inventor and IBM Watson both use deep parsing, only Dr Inventor focuses on analogical comparisons (and conceptual blends). Goel, et al. (2015) discuss how students used Watson first as an aid to co-creativity and subsequently as a means of actively enhancing co-creativity.

KnIT (Spangler, et al., 2014) aims to predict scientific discoveries by analyzing past literature. It extracts and collects information from multiple publications, looking for literal similarities that focus on central topics. KnIT has proposed a novel and testable hypothesis related to a tumor sup-

pressing protein called *p53*. While Dr Inventor and KnIT focus on scientific literature, only Dr Inventor explores non-literal similarities between publications.

Literature Based Discovery (LBD) (Bruza & Weeber, 2008) is also arguably a creative undertaking, whose ABC model seeks knowledge (B) connecting distinct bodies of literature (A and C). CrossBee (Juršič, Bojan, Tanja, & Lavrač, 2012) adopts an LBD-like approach by identifying cross-context terms (not documents) that form connections between publications in distinct areas of research.

Dr Inventor differs from IBM Watson and KnIT by focusing on analogical similarities between ideas. Gentner (1983) distinguishes four categories of similarity (Table 1), highlighting differences between surface features and the deep structure of information. Dr Inventor searches for non-obvious structure-based analogies between publications with few obvious surface similarities (few similar objects).

Table 1, Dr Inventor focuses on Analogical Similarity while deliberately avoiding literal similarity

	Similar attributes and objects	Similar relational structure	Example
Literal similarity	Many	Many	<i>Proxima Centauri is like the Sun</i>
Surface similarity	Many	Few	<i>A candle is like the sun</i>
Analogy	Few	Many	<i>The atom is like the solar system</i>
Dissimilar or Anomaly	Few	Few	<i>The atom is like a chicken</i>

COINVENT (Schorlemmer, et al., 2014) is another concept invention system which attempts to build a formal model of conceptual blending by drawing various interdisciplinary research results. COINVENT is aimed at gaining a deep understanding of conceptual blending and developing a formal method for building a generic creative computational system. COINVENT uses mathematics and music as a working domain, while Dr Inventor focuses on analogical (not literal) comparisons between graphics publications.

Reasons for focusing on analogical similarity (over literal similarity) include: a long-standing view that analogy is an important mode of scientific creativity (Koestler, 1964), existing tools already support literal (but not analogical) similarity, and computational advances in language processing and analogy modelling enable efficient identification of analogies between text-based publications.

Dr Inventor and SIGGRAPH

Dr Inventor is a Creativity Enhancement Tool (CET) operating as a partial simulation of scientists’ creative thinking about research literature. Testing uses a corpus of 1146 papers from the SIGGRAPH¹ conference series² (2002 to 2015). It adopts a task-divided (Kantosalo & Toivonen,

2016) approach to co-creativity, where Dr Inventor searches for creative sources, but evaluating the *usefulness* of its hypothesis (discussed later) is the users’ responsibility.

Reasons for focusing on a specialized domain like computer graphics are, firstly, it ensures a somewhat consistent degree of novelty between publications, allowing any resulting comparisons to be consistently evaluated. Secondly, the resulting inferences should generally be more semantically plausible than might arise from two semantically unrelated papers (say economics and computer graphics). Finally, we may use expert evaluations under the consensual assessment technique (Baer & McKool, 2009), while minimizing any impact from relative expertise in other disciplines.

Previously (Abgaz, et al., 2016-b; Abgaz, O’Donoghue, Smorodinnikov, & Hurley, 2016) explored several metrics measuring the outputs generated by Dr Inventor. This paper builds on that work by exploring new metrics not previously discussed, including several new metrics for mappings and inference that were not previously addressed.

Professional Level *Pro-c* Creativity

Dr Inventor models professional creativity (Pro-c) (Kaufman & Beghetto, 2009) representing “*effortful progression beyond little-c that represents professional-level expertise in any creative area.*” Dr Inventor is a model of analogy-based professional creativity, which might never achieve the eminence associated with Big-C creativity.

Boden’s (2004) refers to the “three main types of creativity – *combinational*, *exploratory*, and *transformational*”³. This paper explores one form of combinational creativity, arising from analogies within the SIGGRAPH corpus. We examine combinational creativity and the space of possible mappings. The central graph-matching problem is NP-hard (Veale & Keane, 1997), with the number of potential target-to-source combinations increasing exponentially with the size of each paper. Multiple source papers each represent a new graph matching problem.

Next, we look at combinational creativity and the inferences. Dr Inventor’s lexical phase identified 26,072 distinct concepts represented as graph nodes (see formal presentation in (Abgaz, et al., 2016-b)) and 7,315 distinct relations (discounting repeated instances), allowing the creation of 4.9×10^{12} distinct combinations of concepts and relations. Furthermore, exploring clusters of just 5 inferences could form 3×10^{63} possible inferences, while the presence of co-references increases the space of possible inferences yet further. By comparison there are around 4×10^{79} atoms in the observable universe.

We believe Dr Inventor also addresses the issue of intentionality (Ventura, 2016) as it only selects those compari-

¹ ACM Special Interest Group on Computer Graphics and Interactive Techniques

² SIGGRAPH is the 5th ranked (of several thousand) conference in computer science by Microsoft Academic Search (accessed February 15, 2017).

³ My emphasis on “combinational”, from Preface to the 2nd Edn.

sons likely to be adopted by an expert user. This paper explores multiple facets of these comparisons to identify those of greatest creative impact. Surprise is also associated with creative comparisons and many of its analogies were found to be surprising, comparing papers from different subtopics that were separated by many years. Dr Inventor’s research hypothesis (page 6) identified “hole” as analogous to “area”, though these terms seem more like opposites.

While Dr Inventor explores combinational creativity, how can it find those analogies that will have the greatest creative impact on professional users? This central question motivates this paper, leading to our initial hypothesis that it is the mapping (or counterpart projection) that is the hallmark of creative comparisons.

The Dr Inventor Computational System

The Dr Inventor system aims to simulate one mode of creative scientific thinking to identify comparisons that might enhance a researcher’s creativity. Dr Inventor achieves this through its deep processing of natural language retrieved directly from research publications, from which it derives an attributed relational graph called a Research Object Skeletons (ROS) in the form (*relation (subject, object)*). Crucially, multiply referenced items are uniquely represented in a ROS (see Figure 1).

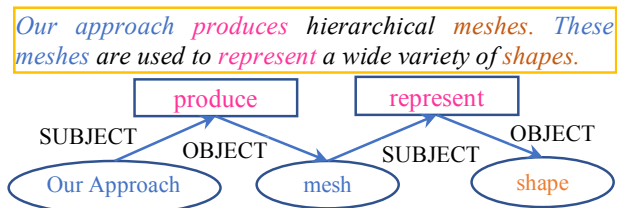


Figure 1. Subject-verb-object triple generated by the graph builder.

Unlike human generated data that neatly segregates mapping data from inference data, our ROS involve a single unified knowledge structure identifying the best mapping between any pair of papers (the Largest Common Subgraph NP-hard problem), typically identify different subsets of these ROS, depending on the particular papers being compared. Any un-mapped information from the source ROS thus becomes available as candidate inferences for possible transfer to the target. A semantic grounding constraint ensures that all accepted inferences overlap, at least in part, with the corresponding mapping. This ensures the inferences relate directly to the identified analogical similarity that effectively forms the justification for those inferences. We believe that identifying the pre-existing similarities and detecting potentially transferrable knowledge should be seen as central parts of the creative challenge - and differentiating between them should not be assumed as part of the inputs to the creative process.

Dr Inventor Architecture: Dr Inventor combines a number of key technologies beginning by extracting the text from a

publication in PDF format using the tools PDFX and Grobid, addressing multi-column layouts, headers, footers, tables etc. Extracted text is passed to the GATE dependency parser (Ronzano & Saggion, 2015), tailored to deal with inline citations and identifying co-referent terms (like “it” and the item it references). Parsing results undergo semantic extraction to identify key information, generating the ROS graphs that represent each publication through the remainder of the cognitive model. Although SIGGRAPH publications frequently include mathematical expressions these formulae are not currently parsed. However, mathematical variables frequently contribute to mappings due to their use throughout documents.

The analogical mapping between two publications are generated from the ROS graphs using a tailored version of the VF2 (Cordella, Foggia, Sansone, & Vento, 2004) sub-graph matching algorithm. VF2 ensures an appropriate balance of two often competing influences, the mapping process of matching semantically similar concepts while simultaneously respecting the different topologies of the input ROS graphs. It restricts the space of possible mappings, allowing *concepts* (noun) to only map with other concepts while *relation* (verb) nodes also map together.

Building on the mapping, Dr Inventor generates the corresponding inferences, representing the expected information created in response to the comparison. Dr Inventor then blends the new information into the pre-existing target and presents it to the user by placing the inferences in the context of the target paper.

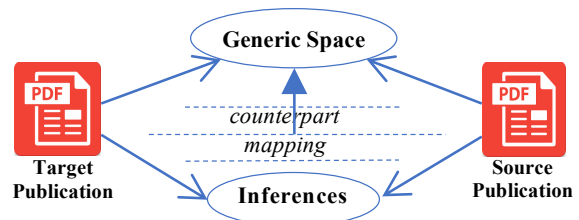


Figure 2: Conceptual spaces used by Dr Inventor, focused on structure driven counterpart mappings

Consistent Terminology

An analogy is a structure-based comparison between two collections of information, centered on a *1-to-1 mapping* between these systems (Gentner, 1983). Later this paper investigates characteristics often associated with the more general cognitive theory of Conceptual Blending (Fauconnier & Turner, 1998). However, the generality of blending can blur the distinction between analogy and literal similarity (Table 1). For simplicity, we define some terminology to clarify that all comparisons discussed in this paper involve structural similarity (and not literal or surface similarity). Where relevant, we use the term *counterpart mapping* to indicate the structure-based 1-to-1 mapping between two ROS conforming to structure mapping theory (Gentner, 1983). This we see as a refinement on blending’s general concept of the counterpart projection.

A mapped pair $P=(S, T)$ is a tuple of source (S) and target (T) items (concept or relational nodes) from the two texts.

The mapping consists of paired items, each identifying the non-literal similarity between them, often being taxonomically related and indicating the Generic Space (Figure 2).

Analogy can be a highly profligate inferential process. To guard against generating many unwarranted inferences, Dr Inventor generates only “grounded inferences” that build directly on information contained in the underlying mapping.

Creative Qualities and Dr Inventor

A user evaluation was undertaken to gather ratings for three qualities of creativity, these being selected from the SPECS list (Jordanous, 2012) as being of greatest relevance to scientific creativity (Abgaz, et al., 2016-b).

Participants: 15 experts in computer graphics were recruited between senior Professors, with many SIGGRAPH publications, to postdoctoral researchers and PhD students having the least experience.

Materials: 10 target publications were chosen using stratified random sampling from across the years of the SIGGRAPH corpus. Dr Inventor explored each possible analogy identifying the best source analog for each target. These analogies were selected based on structural and semantic metrics focused on the counterpart mapping, using the *AnaSim* metric described below.

Evaluators were shown a training video describing analogical comparisons and showing use of the Dr Inventor system. The 10 selected analogies were given to the users who spent approximately 50 minutes exploring each publication before giving their rating evaluations. All evaluations were completed using the Dr Inventor system online⁴ over the course of three days.

Procedure: Users were provided with electronic copies of the source and target paper on the Dr Inventor system. They were asked to read each paper then were asked to explore the identified analogy, a process that was directly supported by presentation of paired terms from the source and target papers. Evaluators were provided with three alternative visualizations of the mapping to support their understanding, allowing users to navigate between the mapped terms and their locations within the documents.

After exploring each comparison user evaluations were collected, also online via a form embedded into the Dr Inventor system. At least 2 user ratings were gathered for each of the following SPECS inspired questions.

- 1) *This is a Novel or Unexpected comparison*
- 2) *This is Potential Useful and Recognizes Gaps in the research*
- 3) *This comparison Challenges the norms in this discipline*

Inter-Rater Agreement: We investigated the agreement between raters for each of these qualities. Our ordinal data coupled with multiple raters required use of Krippendorff (Krippendorff, 2011) inter-rater agreement, returning values between 0.0 and 1.0 with 1.0 indicating maximum agreement. Krippendorff’s alpha for each quality was found to be: *Novelty*= 0.382, *Usefulness*=0.26 and *Challenge the norms*=0.39. While this level of agreement may appear low,

we argue that these creativity ratings are still valid firstly because of the relatively large number of rating categories (5), reducing the alpha score. Additionally, creativity is often seen as highly personal and dependent upon users’ expertise and experience. Post-evaluation discussions highlighted why experts gave very different ratings for a few comparisons, focusing on expertise on specific topics. We note that *Usefulness* showed the lowest agreement with differences in expertise causing disagreement.

Interestingly, more senior users (Professors and Senior Lecturers) found greater value in Dr Inventor than less experienced researchers. Due to the lack of agreement between raters and the fact that each comparison was the best of 1146 (computationally selected) possible comparisons for that target, the normal distribution was seen as inapplicable. Thus, the following evaluations rely on non-parametric statistical techniques.

Counterpart Mapping Metrics

To quantify the degree of similarity existing between any analogous pair, we employed a number of computational metrics focusing on different aspects of a comparison. First, we specify metrics related to the topological similarity, then metrics for semantic similarity and finally a number of combined metrics incorporating both influences. Metrics presented in this section focus on the counterpart mapping, while the semantic similarity relates to the generic space. Both structural and semantic factors form central parts of the VF2 (Cordella, Foggia, Sansone, & Vento, 2004) based (sub-)graph isomorphism mapping algorithm. Additionally, the semantic scores outlined below directly influence the best mapping that is identified between analogs. As noted by (Van Mieghem, 2013) and others there is no single metric that can usefully compare different graph topologies, leading to our multi-parameter investigation.

Of course, any successful metric might be easily adopted as a basis for data mining (Toivonen & Gross, 2015) for the expeditious identification of creative analogies. However, the focus in this paper is on identifying qualities and associated metrics that perform best as indicators of creative comparisons. Martins, Pollak, Urbancic, & Cardoso (2016) discuss general optimality principles for conceptual blending, but do not address creative comparisons.

We now examine each of the metrics to assess its impact on the level of creativity attributed to each analogy, beginning with the topology-based metrics before moving on to the semantically focused ones.

1. Size of the Mapping (*MapSize*): We examined the result for an expected relationship between the size of the mapping and the ratings awarded by users to each comparison. A Spearman rank order correlation $r_s = 0.212$ ($p=0.279$) indicated that mapping size was not an influence in users’ perception of creativity. This was somewhat surprising as larger mappings were expected to be more convincing and thereby promote creative thinking.

⁴ Available from DrInventor.eu

2. Ratio of Mapped Information (*MapRatio*): This counterpart metric quantifies how much of the target graph participates in the mapping. This measure will result in a higher similarity score for targets that have been thoroughly accounted for by the mapping.

$$\text{MapRatio} = \text{MapSize}/\text{TargetSize}$$

MapRatio produces values between 0 and 1 where values near 1 indicate a greater portion of the target problem participates in the mapping.

A Spearman rank order correlation between MapRatio and the average creativity score was calculated but was found to be not significant $r_s = -0.006$ ($p = 0.492$). So MapRatio does not appear to be an important factor in identifying creative comparisons. Again, a somewhat surprising result suggesting that merely re-interpreting a target is not sufficient to cause a creative impact. Of course, MapRatio does not incorporate any measure of the source analog.

3. Jaccard Coefficient (*JCoef*): To measure the mapping in relation to both source and target analogs we use the Jaccard coefficient (Jaccard, 1901), which is a measure of the similarity between two finite sets. In this case, the mapping is treated as the intersection between the source and target graphs (Abgaz, et al., 2016-b). This measure incorporates the size of the mapping together with the source and target graphs. It produces a value between 0 and 1 where values near 1 indicate greater structural similarity (homomorphic).

A Spearman rank order correlation indicated the Jaccard coefficient did not identify creative analogies $r_s = 0.078$ ($p = 0.41$). Another thought provoking result as the proportion of the two analogs that participate in a comparison appears to have no impact on the resulting creativity – bearing in mind the small sample size involved in this paper.

So, metrics that focus purely on the topology of the analogs do not appear to be adequate in identifying creative comparisons. Of course, we need to remain cognizant that this does not mean the mapping is irrelevant to creativity, as this evaluation focused only on the best analogies identified for each target. But perhaps semantic factors will prove more fruitful in our quest for creative analogies.

Generic Space Metrics

Dr Inventor quantifies the semantic similarity between mapped counterparts, which are aligned by the mapping process. The generic space generalizes across each pair of mapped items from the two publications (for structural reasons, these may not necessarily be the most semantically similar pairings). Dr Inventor pays particular attention to mapped but not identical items as semantically distant “between domains” comparisons are often seen as the hallmark of creative comparisons (Koestler, 1964).

In this paper, we estimate semantic similarity using the *Lin* (Lin, 1998) similarity metric, which in turn is based on the WordNet lexical database. The *Lin* metric produces results in the range [0-1] with values closer to 1 indicating a greater degree of semantic similarity.

Dr Inventor maintains independent metrics for noun and verb based similarity for several reasons. Firstly, WordNet’s verb-based entailment hierarchy is generally shallower than

its noun hierarchy and this has implications for the *Lin* metric that incorporates the “lowest common subsumer” in its calculations. This causes potential problems and inconsistencies when comparing the relative influence of noun-based and verb-based differences in comparisons. Secondly, a mapped predicate typically involves two mapped nouns but only one mapped verb, which might easily lead noun based similarity to overwhelm the relational similarity. Finally, structure mapping theory (Gentner, 1983) can help differentiate between “literal similarity” and “analogy” by favoring mappings between unrelated nouns but similar verbs.

1. Conceptual Similarity (*ConSim*): Conceptual Similarity measures the similarity of paired concepts (nouns) using the *Lin* metrics. For example, a boat and a car, $\text{Lin}(\text{boat}\#\text{n}, \text{car}\#\text{n}) = 0.7198$, share more commonality than a boat and a “cat”, $\text{Lin}(\text{boat}\#\text{n}, \text{cat}\#\text{n}) = 0.1647$. Our mapping algorithm selects the pair with a higher similarity score when it is presented with such a choice. For a given analogy, we use the mean value between the nouns involved in the mapping.

A Spearman rank-order correlation coefficient revealed a moderate negative relationship between user ratings and the estimate of conceptual similarity (*ConSim*) $r_s = -0.442$, $p = 0.09$. This was an interesting finding as analogies involve, almost by definition, comparisons between different objects. So, a low (though positive) correlations was quite expected, but this strong negative correlation was quite surprising – despite not quite reaching a level of statistical significance. This finding suggests that creative comparisons don't just pair objects with little similarity to one another - but involve objects that are notably and quantifiably *dis*-similar to one another! This, finding was even more surprising given the semantic homogeneity associated with using papers only from SIGGRAPH.

This finding can be seen as allied to the idea that creative comparisons arise from “between domains” comparisons typically involving dissimilar objects (Blanchette & Dunbar, 2000). This conceptual dissimilarity can also be seen as comparable to factors such as the “tension” associated with between domains comparisons.

2. Relational Similarity (*RelSim*): Relational Similarity measures the similarity of paired relations (verbs) again using the *Lin* metric, ensuring that lexically ambiguous terms are interpreted in their verb sense only. To estimate relational similarity, we use the mean relational similarity, averaged across all mapped relations in the comparison.

A Spearman rank-order correlation coefficient revealed a moderate relationship between user ratings and the estimate of relational similarity (*RelSim*) between analogous publication $r_s = 0.430$, $p = 0.10$. This positive but weak relationship was surprising because we expected it to be even stronger. Relational similarity can be seen as the semantic foundations of systematicity theory (Gentner, 1983).

3. Latent Semantic Analysis (*LSA*): Previous studies have shown that similarity as estimated using LSA is not useful in identifying detailed analogies (Ramscar & Yarlett, 2003) or creative comparisons (Abgaz, et al., 2016-b). A Spearman Rank-order correlation between the average creativity rating and the LSA for each comparison was $r_s = -0.6201$ ($p < 0.05$),

suggesting that increasing the semantic distance between analogs had a positive impact on users' creativity. Our results show that conceptual and relational similarity have very different influences on users' perceptions of creativity.

Inference Metrics (Blended Space)

We next present several metrics related to the inferences and creation of the blended space. Dr Inventor counts the inferences it generates from each comparison, allowing identification of the more creative, if not profligate comparisons.

1. Number of Inferences (*NumInfs*): Finally, we examined the impact of the number of inferences upon the average rating awarded to a comparison. The number of grounded inferences indicates the amount of new information the analogy provides. A Spearman rank order correlation $r_s=0.286$ ($p=0.21$) did not show any reliable influence from the number of inferences on users' creativity.

While the Spearman correlation tests for the presence of linear associations between variables, the Wilcoxon signed rank test looks for differences between two populations. A Wilcoxon paired signed rank test between the number of inferences and the user ratings for each analogy revealed that the null hypothesis could not be rejected ($V=7$, $p<0.05$). Additionally, a Pearson Product Moment correlation of 0.613 was identified. Thus, we infer that the number of inferences had an impact on the creativity ratings awarded by Dr Inventor's users.

2. Novelty of Inferences (*ObservedNovelty*): The raw count of the inferences doesn't address the properties of *novelty* and *quality* that are central to creativity (Boden M. A., 2004). We situate our estimation of novelty within the socially creative context (Corneli, 2016) of recent publications in this conference series. Thus, we assess novelty using an

n-gram approach (Abgaz, O'Donoghue, Smorodinnikov, & Hurley, 2016) derived from the SIGGRAPH corpus, with the resulting *n*-grams estimating the novelty of inferences.

Firstly, a tri-gram model was constructed from the entire SIGGRAPH corpus of 721,301 triples, with 604,873 distinct triples (ignoring duplicates). For example, a common inference was found to be: *we (introduce, algorithm)*, however most inferences were novel with respect to these tri-grams.

The corresponding bi-gram model was constructed allowing "piecemeal" evaluation of the (relative) novelty of unfamiliar inferences. So, the *Subj-Verb*, *Verb-Object* and *Subject-Object* combinations can be evaluated in a piecemeal fashion, allowing Dr Inventor to compare the degree of novelty contained within a novel inference. Lower bigram probabilities arise from truly novel combinations of information, indicating a greater level of creativity. Let *i* signify an individual inference and $|i|$ represent its trigram frequency in the repository, then the *novelty* of the inference, $N(i)$ is given by

$$N(i) = \begin{cases} 0 & |i| > 0 \\ 1 - P(i) & |i| = 0 \end{cases}$$

where $P(i)$ is the bi-gram probability.

For a given analogy producing *m* individual inferences, the novelty score is calculated as the average novelty scores of the individual inferences.

The Spearman rank order correlation between the observed novelty score and the user ratings $r_s=0.42$ ($p=0.11$). While this result was not reliable, partly due to the small sample size, it does suggest that novelty of inferences is a factor in creativity ratings. Thus, we argue that this shows that the novelty of inferences might be a factor influencing users' perceptions of creativity. Further evaluations will be required to explore this factor in greater detail.

A Research Hypothesis by Dr Inventor, by N.C.C.A., Bournemouth University, UK

A research hypothesis created by Dr Inventor's led to the following research project. 'Curve-Skeleton Extraction from Incomplete Point Cloud (2009)' describes an algorithm for curve skeleton extraction from point clouds, where large portions of data are missing during 3D laser scan. Dr Inventor system has identified 'Fast Bilateral Filtering for the Display of High-Dynamic-Range Images (2002)' as analogous, presenting a technique to display high-dynamic-range images, which reduces the contrast while preserving details.

The **creative analogy** sees both papers focus on the reconstruction of hidden structural information. Paper-1 solves a 3D problem of incomplete vertex data containing **holes** (caused by self-occlusions during 3D laser scanning). Paper-2 solves a 2D problem in images with poor light management, with under-exposed and over-exposed **areas**, and light behind the main character.

Proposed solution: The papers are from different Computer graphics domains (Modelling and Image processing) and their methods cannot interpolate with each other. Interestingly, "holes" in the problem are mapped with "areas" in the source paper. Analogous examples from existing work has lead us to the following developments.

Inspired analogy: We have explored new ideas through Dr Inventor to learn; How to rebuild and animate 3D models automatically and reconstruct hidden structure more efficiently. A new idea has been generated after a case-study was carried out from the literature of 18 papers by Dr Inventor. We have found a new method to represent natural flower shape, flower blooming and the decay process, using an Ordinary Differential Equation (ODE)-based surface modelling & simulation technique. Interestingly, the analogy paired "area" with "hole", normally seen as opposites.

Shape representation of flower is challenging and interesting topic which has attracted the many researchers. The shape of flower consists of a multi-layer architecture (petals, stigma, and stems). Each part of a flower involves a complex geometrical deformation such as bend, stretch, shrink and curl. Various techniques (Data-driven, Sketch-based, Point-based and Image-based) are popular, but face challenges such as the geometry of high fidelity and missing-captured data.

Advantages of our new method: In order to address the existing challenges for the shape representation and simulation of flower, we present a single framework which uses ODE-based surface modelling & simulation technique to solve geometry structural information more efficiently. Our method is very useful for 3D modelling and simulation that creates realistic flower shapes with a small data size.

3. Novelty Relative to Other Inference (*PredictedNovelty*): We also compared inferences against all *other inferences* generated from all possible analogies from our corpus. Dr Inventor explored over 1.3 million analogies producing 225,230 inferences, of which 151,200 were unique (ignoring duplicates). We might think of this collection as inferences likely to arise (at least analogically) from the collected wisdom contained in the corpus. Tri-gram and bi-gram models were used to estimate the novelty of inferences in relation to all other inferences. However, a Spearman rank order correlation between the novelty of the SIGGRAPH inferences and the user ratings was $r_s=0.048$ ($p=0.446$) showed that this was not an influencing users' perceptions of creativity.

Multi-Space Metrics

1. Analogical Similarity (*AnaSim*): This evaluates the mapping in terms of structural and semantic factors, combining Jaccard's coefficient with conceptual and relational similarity:

$$AnaSim = ((RelSim + ConSim)/2) * JCoef$$

A Spearman Rank-order correlation between the average creativity rating and *AnaSim* showed $r_s=0.349$ ($p=0.16$). Again, this is suggestive of a mild relationship between *AnaSim* and creativity ratings.

2. Overall Similarity Indicator (*OverallSim*): Finally, we look at a theoretically driven combination of these metrics giving a single usable means of selecting the most creative comparisons. Rather than using the number of metrics we employ an exponential squashing function scaling the number of inferences to the range [0...1] in order to select analogies with a moderate number of inferences, while comparisons offering huge number of inferences will gain little advantage. This intervention was made to avoid overwhelming users with too many inferences.

We combine overall analogical similarity (*AnaSim*) score with the scaled inference metric. Novelty is used to determine whether we should include the inference during the presentation to the users. This metric has been used to order the inferences, allowing users to focus on the more informative ones first, when many are available.

$$OverallSim = AnaSim * numInf$$

Final Evaluation

The influence of individual metrics may differ when used in combination with one another, so we explore linear combinations of these different facets using principal component analysis (PCA) (Jolliffe, 2002). PCA is often used to explain the variance within data and while a few of the metrics above (e.g. *AnaSim*) are already linear combinations of more primitive metrics, this evaluation focuses exclusively on the primitive metrics. A PCA analysis was conducted and results show that the first principal component accounting for 63% of the variance was formed the following combination of factors:

$$-0.533ConSim + 0.529RelSim + 0.485numInf + 0.311PredictedNovelty + 0.289ObservedNovelty + 0.144JCoef$$

The first (and thus largest) principal component indicates (by dint of the -0.533) that more creative comparisons involve conceptual (noun-based) *dis*-similarities, as discussed

earlier. Conversely, relational (verb-based) similarity appears to be a factor in creative comparisons, as are greater numbers of inferences. The two novelty scores are also important factors in this principal component, helping remove analogies suggesting uncreative inferences. Four principal components account for all variance in this collection.

This combination of factors may allow identification of better and even more creative comparisons in future versions of the Dr Inventor creativity enhancement tool.

Conclusions

This paper presents a computational system called Dr Inventor that explores novel analogical comparisons between published research documents. Dr Inventor combines a lexical analysis phase with a model of analogical thinking, which forms the core of our model of analogical reasoning and conceptual blending. This paper focuses on the specific problem of identifying the qualities of creative comparisons that help make them creative. Qualities and associated metrics of comparisons were explored, derived from the mapping (counterpart projection), generic space and blended spaces, focusing on both semantic and topological factors.

The main finding in this paper concerns the characteristics of analogy-based comparisons and their potential use as predictors of creative comparisons. Results suggest that it is not the strength of the analogy (or counterpart projection), but rather it is the inferences and their novelty that are the hallmarks of creativity. In particular inferences and their novelty play a significant factor in the creativity ratings given by expert users of the Dr Inventor system.

Dr Inventor treats all inferences as "additive" to the existing body of knowledge, but adding the ability to detect contradictory beliefs might require an incompatible/alternative "belief space". Dr Inventor might thereby well be extended to support novel and alternative belief spaces of Boden's Transformational Creativity. The main challenge lies in determining greater semantic specificity for its conceptual and relational nodes, requiring advances in language processing, semantic tagging and ontology.

We see Dr Inventor's model of non-literal similarity as being one possible approach to supporting creative reasoning across research disciplines. While Dr Inventor has currently only been tested on documents from SIGGRAPH, we believe it points the way for useful progress. We believe that systems like Dr Inventor may offer vital leverage in promoting inter-disciplinary thinking and research.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme ([FP7/2007-2013]) under grant agreement no 611383.

References

Abgaz, Y., O'Donoghue, D. P., Smorodinnikov, D., & Hurley, D. (2016). Evaluation of Analogical Inferences Formed from Automatically Generated Representations of Scientific

- Publications. Artificial Intelligence and Cognitive Science (AICS). Ireland.
- Abgaz, Y., O'Donoghue, D. P., Hurley, D., Saggion, H., Ronzano, F., & Smorodinnikov, D. (2016-b). Embedding a Creativity Support Tool within Computer. ECAI - Modeling & Reasoning in Context.
- Baer, J., & McKool, S. S. (2009). Assessing creativity using the consensual assessment technique. In Handbook of research on assessment technologies, methods, and applications in higher education (pp. 65-77). IGI Global.
- Blanchette, I., & Dunbar, K. (2000). How analogies are generated: The roles of structural and superficial similarity. *Memory & Cognition*, 28(1), 108-124.
- Boden, M. A. (1996). *Dimension of Creativity* (2nd ed.). Cambridge, Massachusetts: MIT Press.
- Boden, M. A. (2004). *The creative mind: Myths and mechanisms*. Psychology Press.
- Bruza, P., & Weeber, M. (2008). *Literature-based discovery*. Springer.
- Cordella, L., Foggia, P., Sansone, C., & Vento, M. (2004). A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 1367-1372.
- Corneli, J. (2016). An institutional approach to computational social creativity. *Intl. Conf. on Computational Creativity (ICCC)*, (pp. 131-138).
- Fauconnier, G., & Turner, M. (1998). Conceptual integration networks. *Cognitive science*, 22(2), 133-187.
- Gentner. (1983). Structure- mapping: A theoretical framework for analogy. *Cognitive science*, 7(2), 155-170.
- Gentner, D., & Smith, L. (2012). Analogical reasoning. In V. S. (Ed.), *Encyclopedia of Human Behavior* (2nd Ed.). (pp. 130-136). Oxford: Elsevier.
- Goel, A. K., Creeden, B., Kumble, M., Salunke, S., Shetty, A., & Wiltgen, B. (2015). Using Watson for enhancing human-computer co-creativity. *AAAI Fall Symposium*.
- Hurley, D., Abgaz, Y., Ali, H., & O'Donoghue, D. P. (2016). Expert and Corpus-Based Evaluation of a 3-Space Model of Conceptual Blending. *ECAI - Evaluating General Purpose AI*.
- Jaccard, P. (1901). Distribution flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37, 241-272.
- Jolliffe, I. (2002). *Principal Component Analysis* (second edition ed.). New York: Springer-Verlag New York, Inc.
- Jordanous, A. (2012). A standardised procedure for evaluating creative systems. *Cognitive Computation*, 4(3), 246-279.
- Juršič, M., Bojan, C., Tanja, T., & Lavrač, N. (2012). Cross-domain literature mining: Finding bridging concepts with CrossBee. *Proc. 3rd Intl. Conf. on Computational Creativity ICCC*, (pp. 33-40).
- Kantosallo, A., & Toivonen, H. (2016). Modes for Creative Human-Computer Collaboration: Alternating and Task-Divided Co-Creativity. *Intl. Conf. Computational Creativity ICCC*.
- Kaufman, J. C., & Beghetto, R. A. (2009). Beyond big and little: The four c model of creativity. *Review of General Psychology*, 13(1), 1-12.
- Koestler, A. (1964). *The Act of Creation*.
- Krippendorff, K. (2011). Agreement and Information in the Reliability of Coding. *Communication Methods and Measures*, 5(2), 93-112.
- Lin, D. (1998). *An Information-Theoretic Definition of Similarity*. San Francisco, CA, USA: Morgan Kaufmann.
- Martins, P., Pollak, S., Urbancic, T., & Cardoso, A. (2016). Optimality Principles in Computational Approaches to Conceptual Blending: Do We Need Them (at) All? *7th International Conference on Computational Creativity*.
- O'Donoghue, D. P., & Keane, M. T. (2012). A Creative Analogy Machine: Results and Challenges. *Proc Intl. Conf. on Computational Creativity (ICCC)*, (pp. 17-24).
- O'Donoghue, D. P., Abgaz, Y., Hurley, D., & Ronzano, F. (2015). Stimulating and simulating creativity with Dr inventor. *Proc. Int. Conf. on Computational Creativity ICCC*, (pp. 220-227).
- Pinel, F., & Varshney, L. R. (2014). Computational Creativity for Culinary Recipes. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)* (pp. 439-442). New York, NY, USA: ACM.
- Ramscar, M., & Yarlett, D. (2003). Semantic grounding in models of analogy. *Cognitive Science*, 1, 41-71.
- Ronzano, F., & Saggion, H. (2015). Dr. Inventor Framework: Extracting Structured Information from Scientific Publications. In *Discovery Science* (pp. 209-220). Springer.
- Schorlemmer, M., Smaill, A., Kühnberger, K., Kutz, O., Colton, S., Cambouropoulos, E., & Pease, A. (2014). Coinvent: Towards a computational concept invention theory. *Intl. Conf. Computational Creativity*.
- Spangler, S., Wilkins, A. D., Bachman, B. J., Nagarajan, M., Dayaram, T., Haas, P., & Regenbogen, S. (2014). Automated hypothesis generation based on mining scientific literature. *Proc. 20th ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, (pp. 1877-1886). 2014.
- Toivonen, H., & Gross, O. (2015). Data mining and machine learning in computational creativity. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6), 265-275.
- Van Mieghem, P. (2013). Can the topology of two graphs be compared by one number? *Delft University of Technology*.
- Veale, T., & Keane, M. T. (1997). The competence of sub-optimal theories of structure mapping on hard analogies. *IJCAI*, (pp. 232-237.).

Teaching Computational Creativity

Margareta Ackerman¹, Ashok Goel², Colin G. Johnson³, Anna Jordanous³,
Carlos León⁴, Rafael Pérez y Pérez⁵, Hannu Toivonen⁶, and Dan Ventura⁷.

¹Computer Engineering Department, Santa Clara University. ackermanmaya@gmail.com.

²School of Interactive Computing, Georgia Tech. ashok.goel@cc.gatech.edu

³School of Computing, University of Kent. {C.G.Johnson, A.K.Jordanous}@kent.ac.uk

⁴Department of Software Engineering and Artificial Intelligence, Complutense University of Madrid. cleon@ucm.es

⁵ Depto. de Tecnologías de la Información, Universidad Autónoma Metropolitana. rperez@correo.cua.uam.mx

⁶ Department of Computer Science, University of Helsinki. hannu.toivonen@helsinki.fi

⁷ Computer Science Department, Brigham Young University. ventura@cs.byu.edu

Abstract

The increasing popularity of computational creativity (CC) in recent years gives rise to the need for educational resources. This paper presents several modules that together act as a guide for developing new CC courses as well as improving existing curricula. As well as introducing core CC concepts, we address pedagogical approaches to this interdisciplinary subject. An accessible overview of the field lets this paper double as an introductory tutorial to computational creativity.

Introduction

As computational creativity (CC) grows in popularity, it is increasingly being taught in some form as a university-level course. However, at this point, there does not exist any canonical pedagogy nor accepted textbook for the subject; thus it may be useful to begin a discussion on pedagogy, topics, best practices, etc. We feel that a CC course is clearly an important part of the study of (at least) artificial intelligence (AI), and offer some talking points to explain why:

- It provides a method of approaching AI holistically.
- It may appeal to traditionally underrepresented groups in CS and related fields, e.g. female students.
- It is a next step in building more robust intelligent systems.
- It broadens views of constituent parts of intelligence.
- It allows students the opportunity to critically reflect about human and computer creativity (which may help improve their own creativity).
- Being interdisciplinary, it exposes our students to other fields of study, which helps expand their horizons and allows them to experience new perspectives.

To begin the discussion, then, we offer the following as a guide list of learning objectives for a CC course:

- Begin to think about computation in new ways
- Understand central questions and challenges in computational creativity (intentionality, evaluation, abstraction vs. domain specialization, sociality, etc.)
- Understand the difference between mere generation and computational creativity

- Master theoretical and conceptual tools for analysing and discussing creative systems
- Develop and understand one's own creative processes
- Become familiar with the latest advances in CC, and be able to critically discuss current state-of-the-art
- Become able to create CC systems and/or techniques
- Gain the ability to describe, employ and debate methods for evaluation of computational creativity
- Be able to identify appropriate contexts for using CC

In what follows, we offer examples of extant CC systems that can be used as archetypes for introducing key questions in the field; discuss different modeling approaches and how they provide complementary accounts of the theory; address human vs. computational creativity and how both play a role in CC education; discuss some key AI techniques and issues that may be appropriately integrated into a CC course; provide some examples of lessons learned, sample assignments and suggested best practices; and discuss interdisciplinary approaches to CC education.

This paper is aimed at:

- CC researchers who would like to teach computational creativity but may not be sure where to start;
- Those who already teach CC, who would like to expand and further develop their courses;
- Researchers new to CC, but perhaps familiar with AI or related fields, who would be able to use this paper as a starting point to learn about the field.

Introducing Computational Creativity

What is computational creativity, and how should we introduce it to our students? We present a working definition and select several systems that can be introduced in a CC course to illustrate central ideas in the field along with discussion questions targeting these ideas. We discuss several philosophically differing approaches to CC. Lastly, we mention connections between CC and human creativity, which provide another accessible entry point into this field while revealing some of its interdisciplinary facets.

Defining Computational Creativity

Recently the PROSECCO research network has defined computational creativity as “*an emerging field that studies and exploits the potential of computers to be more than feature-rich tools, and to act as autonomous creators and co-creators in their own right. In a CC system, the creative impetus comes from the machine, not the user, though in a hybrid CC system a joint impetus may come from both together.*” (Intro to CC). However, the definition of computational creativity has a complex and argued history, and indeed debates around its definition—or simply the definition of creativity *tout court*—can be an interesting way to engage students in the subject, much as arguments around the nature of intelligence can enlighten a discussion of artificial intelligence. Below we present several alternate approaches to defining CC that can be used to spark this discussion.

Complementary to giving a formal, intentional, definition of CC is to provide examples of systems that have been argued as being creative and to engage students in discussions about why these systems are, or are not, creative. Furthermore, an interesting contrast can be drawn with human creative systems; taking a CC system and a human system with similar types of outputs, and considering whether one is creative and the other not, is an interesting and informative exercise.

Some of the systems that we have found useful—and the CC topics/questions that they are useful in introducing to students—include:

- *The Painting Fool* (Colton 2012). Is it necessary for CC systems to be able to explain their creative decisions? How can a CC system draw on external world knowledge? Does engaging with artworld economics help us value the creativity of a CC system?
- *AARON* (McCorduck 1991). To what degree can a system of parameterised outputs and constraints be considered to act in a creative way? Where does the creativity lie in a system that has been developed over several decades, where the outputs from the system have influenced the development of the system?
- Emotive music generation (Monteith 2012). How important is evoking an emotional response in an audience for an artistic CC system? Does it matter that computers can’t “feel” the emotion? Does a human creative artist need to be able to feel the emotion that they are aiming to engender in their audience?
- Experiments in Musical Intelligence (Cope 1996). What kind of creativity can be generated by analysis and abstraction from a corpus of existing material? Where does pastiche end and creativity begin? What is the role of inspirational examples in CC systems?
- Mexica (Pérez y Pérez 1999). What are the differences between the creativity needed to create structures (e.g. for stories) and that for creating language?
- The Joking Computer (Ritchie and Masthoff 2011). How can something that cannot laugh create jokes? Can a CC system that consists of filling in templates be regarded as creative? Would a system’s outputs be regarded as creative if generated by a child?
- DARCI (Norton, Heath, and Ventura 2013). What role does intention play in creativity? How do perceptual grounding and communicating intention relate?
- IDEAL (Goel and Bhatta 2004) How do designers generate and evaluate design ideas for novel problems? What role does analogical thinking play in idea generation? What role does systems thinking play in evaluating a design idea?
- MILA (Goel and Joyner 2015) How do scientists make new discoveries? And model observed phenomenon? How can we support citizen scientists and student scientists in conducting authentic science?
- Poem Machine (Kantosalo et al. 2014) How can a computer and a human collaborate in co-creating? What does it take to transform a creative system to support human-computer co-creation? Can creative authorship be shared?
- Artificial Creativity (Saunders and Gero 2001) How do societies of creative agents behave? When is creativity an emergent property of a society?
- ER-Model (Pérez y Pérez and Sharples 2001). Can a computer model of creativity be used in multiple domains? What are the differences and similarities between those implementations?

Approaches to Computational Creativity

Several different approaches to computational creativity have been proposed. Here we discuss some notable approaches suitable for introducing CC. In particular, we contrast the operational, product-centered approach with the cognitive perspective that focuses on the creative process. Another popular model, expressing creative methods as a search problem, is also discussed.

Cognitive vs. Engineering Approach One approach to the problem of computational creativity might be characterized as operational, *as the study and simulation, by computational means, of behaviour, natural and artificial, which would, if observed in humans, be deemed creative* (Colton and Wiggins 2012). As Jordanous points out, from this perspective “the challenge is to engineer a system that would be judged to be creative by its audience, rather than engineering a system that possesses a level of creativity existing independently of an audience’s perception” (Jordanous 2012b). In general terms, this kind of approach employs mathematical models and engineering methods.

A second approach might be characterized as an interdisciplinary study of the creative process employing computers as the core tool for reflection and generation of new knowledge (Pérez y Pérez 2015a). This perspective emphasizes the importance of contributing to the understanding of the creative process. It attempts to help answer questions such as, how do we conceive new ideas? How can we produce coherent sequences

of actions during the creative act? How do we assess the quality of an artifact? This approach is motivated by the work of philosophers, sociologists, cognitive psychologists, and so on.

In this way, the engineering/mathematical perspective concentrates on products and processes that would be validated by an audience, while the cognitive point of view privileges the generation of insights about the phenomenon we are studying. We can imagine these ideas as a continuum, on which each of the extremes represents one of these approaches. We refer to it as the CC-continuum. Most of the systems and models that have been developed in recent years can be placed along this continuum. However, we are aware that there are other possible classifications.

We believe that the CC-continuum can be a valuable tool for teaching CC. It provides a wide view of the kind of systems that have been developed and invites students to contrast their core features. In the same way, it might be useful to assist students in developing analytical skills. For instance, each time a new system or model is studied and discussed in the classroom, the students may be asked to locate it within the continuum and justify their reasons.

Creativity as Search Another abstraction for studying and describing creative methods is to view them as search in some space of potential artifacts (Wiggins 2006). According to Wiggins’ framework, a creative system can be defined essentially by three components: 1) a search space of valid artifacts, 2) an evaluation method that indicates whether or not an artifact is valued, and 3) a search method to traverse the search space for valued artifacts.

This provides a relatively simple and generic approach for describing and comparing CC systems. For instance, given two systems that produce stories, how do they differ in what they consider to be valid stories? How do their methods for measuring if a story is good compare? How do they find (generate) good stories?

The conceptual division of creative operations into three distinct components (search space, value, search method) supports discussion on various ways to achieve transformational creativity. A system is *transformationally creative* if it carries out informed changes on its own operation. Transformational creativity is interesting because a system that is not transformational can be argued “to just do what it was told to do”. Using the framework of Wiggins (2006), it is natural to ask if, and explain how, a system possibly modifies its search space, its way of valuing artifacts, or its method of searching for them.

Human and Computer Creativity

Research into computational creativity in the cognitive systems paradigm is related to human creativity much like research into artificial intelligence in the paradigm is related to human cognition (Goel and Davies 2011). This is in part because humans are the recipients of

the products of computational creativity and in part because computational creativity is (at least) indirectly inspired by human creativity.

More generally, this is a two-way relationship. In one direction, theories of human creativity generate hypotheses and models for realizing computational creativity. In the other direction, computational creativity techniques act as hypotheses for understanding creativity in humans. On one hand, theories of human creativity provide constraints for techniques of computational creativity so that the latter are meaningful to the former. On the other, implementation on computers helps evaluate theories of creativity in humans.

A CC course may introduce both the core processes of human creativity and their use in computational creativity. The creative processes addressed may include (1) Design Thinking (thinking about ill-structured and open-ended problems with ill-defined goals and evaluation criteria) (2) Analogical Thinking (thinking about novel situations in terms of similar, familiar situations), (3) Meta-Thinking (thinking about one’s own knowledge and thinking), (4) Abductive Thinking (thinking about potential explanations for a set of data), (5) Visual Thinking (thinking about images and in images), and (6) Systems Thinking (thinking about complex phenomena consisting of multiple interacting components and causal processes) (Goel et al. 2015).

The study of human creativity also offers an accessible entry point for students new to CC, especially for interdisciplinary classrooms that include students whose primary field of study is outside of computer science. The history of human creativity and myths surrounding this concept (such as the idea that children are more creative than adults) (Sawyer 2011) can act as a foundation for reasoning about computer creativity. A summary of psychology research on how humans can become more creative (Sawyer 2013) can further their understanding of creativity while directly strengthening their own creative capacities.

From Generation to Evaluation

It may be argued that the two most basic abilities for any CC system are that of *generation*—the ability to produce something—and *evaluation*—the ability to assess something. Indeed, a large proportion of the scientific contributions to the field has been historically devoted to *generative systems*.¹ As such, it is likely safe to suggest that generative techniques are necessary, if not sufficient, for computational creativity. It then follows that teaching CC should address computational generative methods. While work on evaluation has lagged somewhat behind that on generation, it is equally critical both for any argument of creativity in a particular system and for assessing the development of the field as a whole.

¹An informal review of the papers appearing in *ICCC 2016* yields 21/32 papers ($\approx 66\%$) discussing generation.

Generative Methods

We assert that teaching CC must address the application of (generative) AI programming techniques, but the question of which ones should be considered fundamental assets is less clear. The following list describes the application of some common AI approaches to CC:

- State space search (part of what is commonly known as *GOFAI*) is fundamental in computational creativity. There are many models of creativity that address problems in terms of *spaces* and artifacts, *distance* between concepts and other objects that can be naturally modelled as refined forms of state space search (Wiggins 2006; Riedl and Young 2006).
- Markov chains have been explored in several subfields, most remarkably in music generation (Eigenfeldt and Pasquier 2010; Nierhaus 2009).
- Knowledge intensive systems (schemas, frames, grammars, rules, etc.) are also among the most common techniques, especially in domains which require some level of knowledge management, like narrative (Álvarez, Pérez y Pérez, and Aliseda 2007; Veale 2013; Bringsjord and Ferrucci 2000).
- Genetic algorithms and evolutionary approaches are often used for performing a more informed search (Baydin, De Mantaras, and Ontanon 2012; Unemi 2014).
- Learning/adaptation via statistical methods has also been explored (Bickerman et al. 2010; Maher, Merrick, and Saunders 2008; Toivonen and Gross 2015).

This list is far from exhaustive, and yet it is already long enough to justify more than one course, even if the students are already familiar with some or all of the fundamentals of these techniques. While some treatment of these topics seems necessary for any CC curriculum, covering all of them is likely unnecessary for engaging and teaching the core foundations of the generative methods involved in CC. As discussed throughout this paper, different courses (current and future) tackle the problem from different perspectives.

It may be the case that no single generative approach is sufficient, and it is therefore common to employ heterogeneous methods to obtain an eclectic synergy—sometimes it is the combination of procedures itself which defines potentially creative characteristics of a system. For example, one might combine search, knowledge and evaluation (León and Gervás 2014).

Evaluation as Part of the Creative Process

The ability of a system to observe and assess its own performance is an elementary requirement for any attribution of self-awareness or intent to the system, and therefore a key criterion for creativity. Internal evaluation of its own products not only tells the system how well it is performing but also allows it to change its behavior in an attempt to perform better. A system that

takes more creative responsibility could even change its standards and modify its own goals.

Jennings (2010) gives an accessible account of *creative autonomy*, the ability of a system to evaluate and to change its standard. While evaluation itself obviously is domain and application specific, the role and importance of evaluation is discussed in the literature more generally. Interestingly, the ability to evaluate artifacts relates to social aspects of creativity in several ways. First, though it may sound paradoxical, creative autonomy actually requires a social setting — interaction with other systems seems the most feasible way for a system to obtain information with which it can change its own standards. Second, a system (or an agent) can contribute to other agents and the society as a whole by evaluating artifacts produced by others.

In its simplest form, an internal evaluation function can be used to implement a generate-and-test architecture: produce an artifact, evaluate it, and only output it if it is good enough. This style of operation coincides with some models of human and social creativity, as described by Saunders and Gero (2001).

Evaluation of Creative Systems

As well as playing a role in the creative process itself, evaluation is also a key part of the scientific research process. We evaluate our creative systems so that we can understand what has been achieved as well as what needs further improvement. A number of key evaluation methodologies and approaches exist for evaluating how successful a ‘creative system’ actually is at being creative (listed in date order):

- 2001/2007: Ritchie’s empirical criteria for assessing a system by examining the typicality/atypicality and novelty of its output against 18 formally stated criteria (Ritchie 2007).
- 2008: The Creative Tripod: a system is a potential candidate for being creative if it demonstrates *skill, imagination and appreciation* (Colton 2008).
- 2011: The FACE model: provides an abstract framework for qualifying different kinds of creative acts performed by a system, both at the product and process level. These acts include Framing information, Aesthetic measures, Concepts, and Expressions (of a concept) (Colton, Charnley, and Pease 2011).
- 2012: SPECS: the Standardised Procedure for Evaluating Creative Systems methodology requires the evaluator to evaluate the system based on standards or criteria that are drawn from a definition/characterisation of creativity that the system is intended to implement (Jordanous 2012a).
- 2004/2015: Pérez y Pérez suggests that, besides being able to generate novel, coherent and interesting (or useful) products, a creative agent must fulfill the following constraints: 1) Employ a knowledge-base to build its outcomes; 2) Interpret its own outputs in order to generate novel knowledge that is useful to pro-

duce more original pieces; 3) Evaluate its own products; such an evaluation must influence the way the generation process works (Pérez y Pérez and Sharples 2004; Pérez y Pérez 2015a).

- 2016: Ventura discusses various thresholds for differentiating mere generation from true creativity and proposes a spectrum of abstracted prototype CC systems that can be used as landmarks by which specific CC systems can be evaluated for their relative creative ability (Ventura 2016).

It is important that students are given opportunities to practice using evaluation frameworks to analyse creative systems, alongside practical work in making the systems themselves. For example, a practical lab class can be set where students try out different evaluation techniques on existing CC systems. For an end-of-module project assignment in which students produce a CC system, they could be required to evaluate their system in a methodical way, such as one of those listed above.

Projects and Assignments

It is fitting for a CC course to include assignments that not only reinforce the material but also help students develop their own creative capacity. To this end, we present several assignments that have been successfully applied in CC courses, with the hope that such experience will simplify design of future curricula.

- **TweeterBots:** Used as an introductory assignment, the design of a simple TweeterBot allows students to engage with several aspects of CC early in the course. The objective of this assignment is to design a bot such that at least some of its tweets could have been made by a reasonably creative person. Achieving two distinct objectives, the creation of a bot gives students a chance to engage their creative capacities when coming up with a novel concept for a bot, while on the technical side allowing them to start delving into generative methods. It is also an excellent demonstration of *mere generation*, which students are asked to surpass later in the course.
- **Art Conversion:** It is often argued that creativity, as found in the arts, aims to express an artist's emotions. The art conversion assignment challenges this notion by asking students to extract the emotional content of artifacts in one art form, and then use this content to create art in a different medium. For example, students may perform sentiment analysis on a poem, then generate music that aims to capture the same feelings as the poem. Similarly, one may begin with a melody, and create visual art that expresses the same mood. This assignment asks students to consider what makes creative artifacts impactful on art consumers, and how computer systems may incorporate this facet of creativity. An evaluation phase may also be integrated into this assignment.
- **Computational Model of Creativity:** Students are asked to produce a high-level computational model of the creative process, focusing on the big picture and the computational aspects of the model, and accounting for as much of the "theory" as possible (i.e. they should say *what* they think the model should include and *why* they think that way and *how* they might actually do that computationally). This is typically assigned within the first week of the semester, to give the students the chance to wrestle with difficult questions before they've been exposed to the current literature.
- **Paper Debate:** Students are given a paper to read ranging in content across various CC debates and applications and some students are selected to informally present their thoughts on the paper, to start off discussion. To help give a rounded view of the paper, some students are asked to present the good points of the paper and others to critique weaknesses of the paper (though we are flexible in whether students actually stick to their assigned 'role' here!) Following the brief presentations, the students critique the paper and the issues it raises, in a group discussion. Students are assessed on the level of critical thinking in their presentations and the quality of their contributions to discussions.
- **Large, Self-proposed Projects:** Requiring students to design and build a major project over the course of an entire semester (possibly as a member of a team) teaches them about inquiry in the processes of both system design and scientific discovery. Such projects require not only eventual solution of design problems but also initial inquiry to understand the design problems. A significant part of creativity lies in this initial inquiry that entails question-answering, problem understanding, and problem formulation—before problem solving commences. By developing semester-long projects that incorporate this philosophy of design inquiry students are led not only to the answers to some questions but also to the formulation of new questions.
- **Project Assessment:** A useful way of integrating several aspects of computational creativity is requiring students to describe and assess their own projects using the concepts covered during the course. For instance, how could the project be described using the FACE model (Colton, Charnley, and Pease 2011) or Wiggins' model of creativity as search (Wiggins 2006)? Which creative acts does it perform? Is it transformationally creative? And so on. Such an assessment task links abstract conceptual issues to concrete implementations, helping students learn more about both. It is important to announce this assessment task at an early phase of the CC course, to motivate learning of the more conceptual material and to encourage students to develop their projects in more creative directions. The project assessments made by students can be leveraged when grading, in

assessment of both the project and conceptual skills.

A project could be made more concrete by asking students to select one or more specific generation methods, an application domain, and a formal evaluation metric amongst those introduced throughout the course. These three components could then be used as a starting point for designing a comprehensive course project.

One assessment method that has proven less successful is a traditional end-of-module examination. Particularly because CC is an area that often allows students to explore any domain of interest (complementing core CC topics), we have found that it is less appropriate to ask students to review an entire course worth of content for an exam scenario; and it is difficult to set exam questions that evenly cover a CC module's content. Many of the assessments listed can be successful alternatives for midterm/finals based around students investigating topics of their choice. Other possibilities for an exam include giving the students selected papers to read and critique during an exam or a take-home exam asking them to answer questions about the development and analysis of systems they have built during the course (possibly using the format of an *ICCC*² paper type).

Interdisciplinary Approaches

Computational creativity is inherently interdisciplinary. This sub-field of artificial intelligent intersects directly with many other fields, including psychology, cognitive science, mathematics and engineering, to name a few, and indirectly with any number of application domains, from musical composition to the culinary arts to scientific discovery. A CC course that combines perspectives from different disciplines can offer an eye-opening experience to its students, by letting them see the world from a different point of view.

Students tend to choose their field of study at a young age, often while still in their teens, and henceforth find themselves in learning environments with homogeneous perspectives and values. Interaction with faculty or students with fundamentally different mindsets not only expands their horizons but also can lead them to questions and opportunities that only arise when two or more fields are combined. Even for classes containing students from a single discipline, they often bring knowledge from personal creative hobbies, helping them relate to the course content on an individual level.

Interdisciplinary connections can be formed with application domain experts (e.g., artists, musicians, dancers, poets) by inviting speakers to discuss their field of expertise, their views on creativity and connections between their field and computing. Assignments can be focused on teaching CC concepts through applications in the relevant domain (e.g., visual art, music, dance, poetry). Projects bringing together students across diverse disciplines (such as computing and dance) has led to original research (Brockhoeft et al. 2016).

²The *International Conference on Computational Creativity* (ICCC) is the primary conference in the field.

Continuing the idea further may involve co-teaching a course with faculty from other disciplines. Different disciplines use their own languages to communicate often distinct goals, and building a common base of understanding can be challenging. For example, in a class of CS and MFA students, co-taught by CS and Visual Arts faculty, the computer scientists could become frustrated that the artists do not think algorithmically (nor perhaps even understand the term) while the artists could believe that discussing art from an algorithmic perspective entirely misses the point at best and completely ruins the concept at worst. However, being placed in an environment that requires effort to understand new ways of thinking broadens the perspective of both sets of students and can result in a successful interdisciplinary experience (Norton, Heath, and Ventura 2011).

Another variant of the interdisciplinary classroom includes students from multiple fields (e.g., architecture, engineering, computing and business) working together on an interdisciplinary project, for which their various skills complement each other. The inclusion of interdisciplinary students in teams results in more authentic/impactful projects, such as recent interdisciplinary projects on computational creativity in biologically inspired design (Goel et al. 2015).

An interdisciplinary classroom in any of its forms offers a variety of challenges, though, as discussed above, the rewards are often worth the effort. To help navigate this challenge, Pérez y Pérez (2015b) describes six features for interdisciplinary work, which apply to the interdisciplinary CC classroom. The first principle addresses the need for awareness of how our disciplinary training shapes our skills and vision of the world, allowing us to compare them with those of others. The second feature addresses the need for a common vocabulary, which makes it possible to communicate across distinct disciplines. Third is the development of academic empathy, that is, the capacity for seeing a problem from the (methodological and epistemic) perspective of others. Fourth is developing trust. Fifth is a willingness to confront ideas and to reach consensus (particularly as related to methodological and epistemic issues). The final feature is good leadership. The leader's main task is to clearly demarcate the roles and responsibilities of people from different disciplines, which can otherwise be confusing, or even intimidating, when working in an interdisciplinary setting. This framework allows students from diverse academic backgrounds (engineers, mathematicians, physicist, psychologists, philosophers, anthropologists, etc.) to analyze different CC models and discuss them from a variety of viewpoints.

Discussion and Conclusions

This paper is written from the perspectives of a group of computational creativity researchers who teach various CC modules to university students from around the world. We intend this paper to act as a guide for those who would like to start to teach CC, or those who already teach CC and would like to review their teaching.

The overarching aim of this paper is to assist people in teaching CC, and to help simplify the process of putting together a CC course. We outline several areas we feel are necessary for such an endeavor; as such, this paper also functions as a guide to computational creativity for someone who is new to the area but familiar with related fields such as AI.

We begin by outlining several learning objectives that would help to shape and guide a new module in computational creativity. Then, we consider how to address the question ‘what is computational creativity’ with students. We consider definitions of CC, highlight some notable CC systems to cover as examples, and identify different ways to approach computational creativity. Again with the view of identifying key content to teach CC students, we also discuss the roles of generation and evaluation in computational creativity.

For educational purposes, assignments and projects play valuable roles in helping our students learn. From our experience teaching computational creativity, we outline a variety of assignment types that have worked well. We note that assignments can not only help reinforce the taught material, but also help our students develop their own creative capacity.

To conclude, the process of sharing our experience leads us to some interesting overarching observations about teaching computational creativity. CC is a highly interdisciplinary area of research, and we reflect on the implications of this for our teaching. Students react to the interdisciplinary nature of computational creativity in different ways; this depends on their ability to adapt to course content in which there are philosophical as well as practical issues to be dealt with, and perhaps no ‘right answers’ to be found. Many of our courses are taught mainly to computing students. Coming from a background of being taught objective material: facts, algorithms, formulae, programming, etc, the students must adapt to the subjectivity of computational creativity, just as an art student would need to adapt to objective approaches to creativity. In teaching to students from any discipline, we need to remain mindful of the open-mindedness that is necessary for students to more deeply appreciate the depth of CC content, outside of what they are used to from their own disciplines.

During a debate on teaching computational creativity at the *Seventh International Conference on Computational Creativity* in Paris in 2016, we discussed how we could share resources for CC education across the research community. For example, we may find it easy to write lectures and exercises on our specialist areas of computational creativity but not have the same depth of specialist knowledge for areas slightly removed from our own research interests. The Association for Computational Creativity website already hosts plentiful useful resources on CC research; it was, therefore, suggested during the 2016 discussion as a good option for a central repository for sharing teaching resources as well. We created a page (<http://computationalcreativity.net/home/teaching-cc/>) dedicated to this purpose,

and hope that this resource will grow as computational creativity education continues to expand.

References

- Álvarez, M.; Pérez y Pérez, R.; and Aliseda, A. 2007. A Generative Grammar for Pre-Hispanic Production: The Case of El Tajín Style. In Cardoso, A., and Wiggins, G. A., eds., *Proceedings of the 4th International Joint Workshop on Computational Creativity*, 39–46. London, UK: Goldsmiths, University of London.
- Baydin, A. G.; De Mantaras, R. L.; and Ontanon, S. 2012. Automated generation of Cross-Domain analogies via evolutionary computation. In Maher, M. L.; Hammond, K.; Pease, A.; Rafael Pérez; Ventura, D.; and Wiggins, G., eds., *Proceedings of the Third International Conference on Computational Creativity*, 25–32.
- Bickerman, G.; Bosley, S.; Swire, P.; and Keller, R. M. 2010. Learning to Create Jazz Melodies Using Deep Belief Nets. In Ventura, D.; Pease, A.; Pérez y Pérez, R.; Ritchie, G.; and Veale, T., eds., *Proceedings of the International Conference on Computational Creativity*, 228–237. Lisbon, Portugal: Department of Informatics Engineering, University of Coimbra.
- Bringsjord, S., and Ferrucci, D. a. 2000. Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a Storytelling Machine. *Computational Linguistics* 26(4):642–647.
- Brockhoeft, T.; Petuch, J.; Bach, J.; Djerekarov, E.; Ackerman, M.; and Tyson, G. 2016. Interactive augmented reality for dance. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Colton, S., and Wiggins, G. A. 2012. Computational creativity: The final frontier? In *Proceedings of the 20th European conference on artificial intelligence*, 21–26. IOS Press.
- Colton, S.; Charnley, J.; and Pease, A. 2011. Computational Creativity Theory: The FACE and IDEA Descriptive Models. In *Proceedings of the 2nd International Conference on Computational Creativity*, 90–95.
- Colton, S. 2008. Creativity versus the Perception of Creativity in Computational Systems. In *Proceedings of AAAI Symposium on Creative Systems*, 14–20.
- Colton, S. 2012. The Painting Fool: Stories from building an automated painter. In McCormack, J., and D’Inverno, M., eds., *Computers and Creativity*. Berlin, Germany: Springer-Verlag. 3–38.
- Cope, D. 1996. *Experiments in Musical Intelligence*. A-R Editions.
- Eigenfeldt, A., and Pasquier, P. 2010. Realtime Generation of Harmonic Progressions Using Constrained Markov Selection. In Ventura, D.; Pease, A.; Pérez y Pérez, R.; Ritchie, G.; and Veale, T., eds., *Proceedings of the International Conference on Computational Creativity*, 16–25. Lisbon, Portugal: Department of Informatics Engineering, University of Coimbra.
- Goel, A., and Bhatta, S. 2004. Use of design patterns in analogy-based design. *Advanced Engineering Informatics* 18(2):85–94.
- Goel, A., and Davies, J. 2011. Artificial intelligence. In Sternberg, R., and Kauffman, S., eds., *Handbook of Intelligence*. Cambridge University Press, 3 edition. chapter 23, 468–484.

- Goel, A., and Joyner, D. 2015. Impact of a creativity support tool on student learning about scientific discovery processes. In *Proceedings of the Sixth International Conference on Computational Creativity*.
- Goel, A.; Creeden, B.; Kumble, M.; Salunke, S.; Shetty, A.; and Wiltgen, B. 2015. Using Watson for enhancing human-computer co-creativity. In *Proceedings of the AAAI 2015 Fall Symposium on Cognitive Assistance*.
- <http://prosecco-network.eu/introduction-computational-creativity>. ProSecco Network website; visited January 2017.
- Jennings, K. E. 2010. Developing creativity: Artificial barriers in artificial intelligence. *Minds and Machines* 20(4):489–501.
- Jordanous, A. 2012a. A Standardised Procedure for Evaluating Creative Systems: Computational Creativity Evaluation Based on What it is to be Creative. *Cognitive Computation* 4(3):246–279.
- Jordanous, A. 2012b. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4(3):246–279.
- Kantosalo, A.; Toivanen, J. M.; Xiao, P.; and Toivonen, H. 2014. From isolation to involvement: Adapting machine creativity software to support human-computer co-creation. In *Proceedings of the Fifth International Conference on Computational Creativity, June 10-13, 2014, Ljubljana, Slovenia*, 1–8.
- León, C., and Gervás, P. 2014. Creativity in Story Generation from the Ground Up: Non-deterministic Simulation driven by Narrative. In *5th International Conference on Computational Creativity, ICCO 2014*.
- Maher, M. L.; Merrick, K. E.; and Saunders, R. 2008. Achieving Creative Behavior Using Curious Learning Agents. In Ventura, D.; Maher, M. L.; and Colton, S., eds., *AAAI Spring Symposium: Creative Intelligent Systems'08*, Technical Report SS-08-03, 40–46.
- McCorduck, P. 1991. *Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. W. H. Freeman & Co.
- Monteith, K. P. 2012. *Automatic Generation of Music for Inducing Emotive and Physiological Responses*. Ph.D. Dissertation, Provo, UT, USA.
- Nierhaus, G. 2009. *Algorithmic composition: Paradigms of automated music generation*.
- Norton, D.; Heath, D.; and Ventura, D. 2011. An artistic dialogue with the artificial. In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, 31–40. New York, NY, USA: ACM.
- Norton, D.; Heath, D.; and Ventura, D. 2013. Finding creativity in an artificial artist. *Journal of Creative Behavior* 47(2):106–124.
- Pérez y Pérez, R., and Sharples, M. 2001. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence* 13(2):119–139.
- Pérez y Pérez, R., and Sharples, M. 2004. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-based systems* 17(1):15–29.
- Pérez y Pérez, R. 1999. *MEXICA: A Computer Model of Creativity in Writing*. DPhil dissertation, University of Sussex.
- Pérez y Pérez, R. 2015a. A computer-based model for collaborative narrative generation. *Cognitive Systems Research* 36:30–48.
- Pérez y Pérez, R. 2015b. Reflexiones sobre las características del trabajo interdisciplinario y sugerencias sobre cómo fomentarlo en el aula universitaria. In *En Vicente Castellanos (Ed.) Estudios Interdisciplinarios en Comunicación. México D. F.: UAM Cuajimalpa.*, 33–50.
- Riedl, M., and Young, M. 2006. Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing* 24(3):303–323.
- Ritchie, G., and Masthoff, J. 2011. The STANDUP 2 interactive riddle builder. In Ventura, D.; Gervas, P.; Harrell, F.; Maher, M.; Pease, A.; and Wiggins, G., eds., *Proceedings of the Second International Conference on Computational Creativity*, 159.
- Ritchie, G. 2007. Some Empirical Criteria for Attributing Creativity to a Computer Program. *Minds and Machines* 17:67–99.
- Saunders, R., and Gero, J. S. 2001. Artificial creativity: A synthetic approach to the study of creative behaviour. *Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney* 113–139.
- Sawyer, R. K. 2011. *Explaining creativity: The science of human innovation*. Oxford University Press.
- Sawyer, K. 2013. *Zig Zag: the surprising path to greater creativity*. John Wiley & Sons.
- Toivonen, H., and Gross, O. 2015. Data mining and machine learning in computational creativity. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5(6):265–275.
- Unemi, T. 2014. Automated Daily Production of Evolutionary Audio Visual Art — An Experimental Practice. In Colton, S.; Ventura, D.; Lavrač, N.; and Cook, M., eds., *Proceedings of the Fifth International Conference on Computational Creativity*, 33–37.
- Veale, T. 2013. Less Rhyme, More Reason: Knowledge-based Poetry Generation with Feeling, Insight and Wit. In Maher, M. L.; Veale, T.; Saunders, R.; and Bown, O., eds., *Proceedings of the Fourth International Conference on Computational Creativity*, 152–159.
- Ventura, D. 2016. Mere generation: Essential barometer or dated concept? In Pachet, F.; Cardoso, A.; Corruble, V.; and Ghedini, F., eds., *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016)*, 17–24. Paris, France: Sony CSL.
- Wiggins, G. A. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19(7):449–458.

Early-creative behavior: the first manifestations of creativity in a developmental agent

Wendy Aguilar¹ and Rafael Pérez y Pérez²

¹Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, UNAM, Mexico City.
weam@turing.iimas.unam.mx

²Departamento de Tecnologías de la Información, UAM Cuajimalpa, Mexico City.
rperez@correo.cua.uam.mx

Abstract

In this position paper we are interested in studying the genesis of the creative process. We suggest that the field of developmental robotics might be useful towards this end. That is, artificial agents that start with basic knowledge and abilities, and that through the interaction with their environment they develop new skills of incremental complexity. With this purpose, we built our developmental agent which implements a computational model for early cognitive development, based on the Engagement-Reflection model of the creative process. This model is named *Dev E-R* (Developmental Engagement-Reflection). In the tests we have performed, the agent can learn the first abilities related to vision and touch. Can these first acquired behaviors be considered as creative? We claim that they can as long as they fulfill the criteria of novelty, utility, emergence, adaptation and motivation.

Introduction

Computational Creativity (CC) is the interdisciplinary study of the creative process employing computers as a core tool for reflection and generation of new knowledge (Pérez y Pérez, 2015). Most of the systems that one finds in the literature describes agents able to paint, to develop narratives, to compose music, and so on; that is, agents that employ representations of (domain-specific and general) knowledge as well as skills to perform their tasks. Surprisingly, it is hard to find research projects that focus on studying the genesis of the creative process. That is, computer models that illustrate how an agent that starts with basic knowledge and abilities can develop, through the interaction with its environment, the skills required to perform creative tasks. For example, behaviors needed to solve problems, such as the ability to differentiate between means and ends. We believe that this is an important area that has not received the attention that it deserves. We refer to it as early-creative behavior.

The field of developmental robotics might be useful towards this end. This area of research is interested in creating artificial agents (physical or simulated) that can learn and develop new skills of increasing complexity, in an open-ended fashion. These kind of works usually takes inspiration from psychological, neurological and evolutionary biological theories about how human's intelligence grows. In (Guerin,

2011a) the reader can find a review of these kind of systems. However, none of them considers the creative point of view involved in cognitive development.

On the other hand, several authors have described different features associated to creative activities (e.g Amabile and Collins, 1999; Cohen, 1989; Steels, 1990). We believe that the first step towards studying early-creative behavior is to establish a criteria to evaluate if the conduct showed by developmental agents might be associated to creativity.

Thus, in this position paper we argue that:

- CC requires to study the genesis of the creative process.
- The field of developmental robotics can provide a useful framework towards this aim.
- It is necessary to establish metrics that help to analyze if a developmental agent is showing a behavior that might be classified as early-creative.

To establish an adequate metric is a complex problem that requires the efforts of the whole community. In this paper, based on our work on DevE-R (Aguilar and Pérez y Pérez, 2015), a developmental agent, we would like to suggest some criteria that might help as starting point. We will use our model to illustrate how such criteria might be employed.

The paper is organized as follows. Section 2 describes the works we used as a base for developing our agent and our evaluation criteria; Section 3 describes our model; Section 4 shows some tests we performed and the results we obtained; section 5 discusses how such a results can be linked to the proposed criteria; Section 6 provides some conclusions.

Theoretical foundations

Dev E-R is a computational model which simulates early cognitive development inspired by the theories of Cohen (1989) and Piaget (1952). On one hand, Leonora Cohen describes creativity as a series of adaptive behaviors in a continuum set of seven levels of development. Initially, creativity involves the adaptation of the individual to their surroundings, and in the higher levels, it involves adapting the world to the individual. On the other hand, Piaget describes adaptation as the interaction of two inseparable process called assimilation and accommodation. Assimilation refers to the way in which a child transforms new information so that it makes sense within their existing knowledge

base, while accommodation happens when a child changes his or her cognitive structure in an attempt to understand new information. Thus, our approach consists in seen cognitive development from Cohen’s perspective, that is, as a creative activity. To capture this idea, we used and extended the computational model of the creative process called Engagement-Reflection (Pérez y Pérez and Sharples, 2001; Pérez y Pérez, 2007). In (Aguilar and Pérez y Pérez, 2015), we presented the results when the agent was only able to see its world, but could not touch it. In this paper we present the results obtained when the agent can touch but not see its environment, and when it could both see and touch. Our main interest is to see which new skills could arise as a result of modifying these sensory abilities, and to discuss whether these can be consider as early creative behaviors.

For the latter objective, based on the work of (Pérez y Pérez and Sharples, 2004; Piaget, 1952; Steels, 1990; Amabile and Collins, 1999) we suggest the following evaluation criteria (see Aguilar and Pérez y Pérez (2014)):

Novelty: A behavior is considered novel if it did not exist explicitly in the initial database of knowledge of the agent (Pérez y Pérez and Sharples, 2004).

Utility: A behavior is considered useful if its serves as basis for the construction of new knowledge that gradually leads the agent to acquire new skills that are typical of the following stage of development. In this context, it is very important that the acquired knowledge be available to be used for subsequent creations.

Emergence: According to the definition given by Steels (1990), our proposal is to consider a behavior has emerged when its origin may not be traced back directly to the components of the system, but rather, is the result of the way in which such components interact with each other.

Motivation: Amabile and Collins (1999) distinguished two types of creativity: 1) intrinsically-motivated and 2) extrinsically-motivated. The first one refers to a behavior that is motivated by internal rewards (e.g., when a child is interested in a task, or finds it satisfactory, or considers it as a personal challenge that he wants to overcome), while extrinsic motivation is focused on external rewards, appraisal or avoiding punishment. Our statement concerning this paper is that a behavior developed by an agent should be considered as creative only if it appeared as a result of an intrinsic or extrinsic motivation.

Adaptation: The ability to adapt ourselves to our environment has been traditionally deemed (probably since Darwin) as a condition needed for the truly creative behavior (Runco, 2007). Additionally, adaptation and creativity are so much related to one another, that even LeoNora Cohen thinks of adaptation as the closest synonym of creativity (Runco, 2007, p.44), who describes the latter as a series of adaptive behaviors in a continuum set of seven levels of development. In Piaget’s theory, adaptation is defined in terms of the processes of assimilation and accommodation.

The developmental artificial agent, and the Dev E-R model

This section provides a brief summary of the description of the virtual agent, as well as the Dev E-R model. For details see (Aguilar and Pérez y Pérez, 2015).

Virtual World

The agent interacts with a 3D virtual world that recreates ordinary places, such as a living room or a playroom. The world contains simple 3D models of typical objects that may be found in real life. For instance, Figure 1 (a) shows a virtual environment consisting of a house filled with furniture, plants, toys, etc., and Figure 1 (b) represents a study room with a sofa, some toys, chairs, and a shelf.

Physical Features

The agent is implemented as a virtual character (see Figure 1c). It can move its head and its hand (see Table 1). It also implements simulated vision and touch sensors, which are used to capture visual and tactile information of the world with which it interacts. The tactile sensor is placed on the palm of its right hand and is implemented as a presence-detecting sensor (determining if a spatial intersection is present between the 3D model representing the hand and any of the objects present in the surroundings).

Name of the action	Description
<i>MHLeft</i>	Moves head left
<i>MHRight</i>	Moves head right
<i>MHUp</i>	Moves head up
<i>MHDown</i>	Moves head down
<i>MHRightUp</i>	Moves head up and right
<i>MHRightDown</i>	Moves head down and right
<i>MHLeftUp</i>	Moves head up and left
<i>MHLeftDown</i>	Moves head down and left
<i>close_hand</i>	Closes hand
<i>open_hand</i>	Opens hand
<i>HandLeft</i>	Moves hand left
<i>HandRight</i>	Moves hand right
<i>HandUp</i>	Moves hand up
<i>HandDown</i>	Moves hand down
<i>HandForward</i>	Moves hand front
<i>HandBackwards</i>	Moves hand back
<i>random_external_action</i>	Randomly picks one of the foregoing actions.

Table 1: Initial repertoire of physical actions, also named “external actions”, that may be executed by the agent.

Cognitive Features

The agent implements five cognitive features: 1) capacity to “see” and “touch” the world around it; 2) simulating an attention process; 3) simulating affective responses, emotional states and intrinsic motivations pushing it to act; 4) has memory; and (5) simulates a process of adaptation to its environment.



Figure 1: Two examples of virtual worlds with which the agent may interact, (a) a living room, (b) a study room; and (c) the developing agent.

The agent can “see” its world Inspired by Piaget’s theory we implemented our agent in such a way that, when it starts operating, it sees the objects that come into its vision scope as luminous spots, whether static or moving, which are detected from data captured by its simulated visual sensor.

The blurs detected are used to create an internal representation of what the agent is seeing. This representation is termed *Current-Visual-Context*, which includes, among others, the features of the objects seen (e.g. its size and color).

The agent can “touch” its world The agent is capable of touching the world around it through the use of a tactile sensor, which is used to detect: 1) the presence of an object in contact with the palm of the hand (can only touch one element at a time) and 2) its texture. In this paper, we are assuming that the agent has learned to recognize a number of textures, which have been labeled as: “ t_1 ”, “ t_2 ”, “ t_3 ”, etc. The object detected is then used to create an internal representation of what the agent is touching. This representation is termed *Current-Tactile-Context*.

The agent simulates an attention process The agent simulates an attention process which it uses to select which object to interact with from the ones detected. This process takes three criteria into consideration. First, the agent is pre-programmed to have preferences. Thus, it prefers to interact with moving objects over the static ones, and it also prefers the ones with bright colors over the ones with dark colors. Secondly, the agent finds novel objects more attractive than older ones. And third, the agent prefers the objects which it has established an affective response to, an emotional state or a certain motivation, as explained in the next subsection.

The agent simulates affective responses, emotional states and motivations that push it to act The agent simulates affective responses, emotional states and motivations which are inspired by Piaget’s ideas, associated with the relation between affectivity and development of intelligence. The first responses consist of intensity and valence, represented by the agent through variables that span along a scale of -1, +1, +2, wherein -1 represents disliking and +1/+2 represent two degrees of liking. The rest are represented internally as Boolean variables having a true value when the agent presents such emotional states or motivations.

The agent has a memory The agent has a memory wherein it stores all its knowledge. Particularly, the agent stores in this memory its current perception of the world (represented through the *Current-Context* structure) and

how it interacts with that environment (represented through schemas).

Current-Context

The *Current-Context* is a structure composed by two parts (see Figure 2a): 1) a *Current-Visual-Context* or a *Current-Tactile-Context*, and 2) the agent’s current expectations, which are defined as an *Expected Current-Visual-Context* or an *Expected Current-Tactile-Context*. In turn, the first ones are composed by two parts (refer to Figure 2b): 1) the features of the object that is in the center of attention of the agent (its color, size, movement and position within the visual field, or the status of the agent’s hand whether open or closed and the texture of the object that the agent is touching at the time); and 2) the affective responses, emotional states and motivations triggered in the agent by such object. With the purpose of providing a simpler, more compact notation, henceforth, current contexts composed solely by affective responses are herein represented in the form of Figure 3.

Schemas

Schemas as used herein are knowledge structures representing the sensory-motor schemas described by Piaget. There are two types: basic and developed.

Basic schemas represent innate behaviors and tendencies observed by Piaget in babies, which are present in the agent from its initialization. These are represented as contexts associated to actions (see Figure 4a). Figure 4b shows an illustration of a basic schema, which associates the situation of feeling disliking, triggered by an object of any color = a , of any size = b , with any movement status = c and in any position within the visual field = d , with the action of performing a random external action.

Developed schemas are constructed based on the interactions of the agent with its environment, and represent new behaviors. These are composed by a context, an action to be executed, an expected context and a set of contexts with which the expectations have been fulfilled (named “Contexts Expectations Fulfilled”), and others that were not fulfilled (termed “Contexts Expectations NOT Fulfilled”), as illustrated in Figure 5a. Figure 5b shows an example of a developed schema.

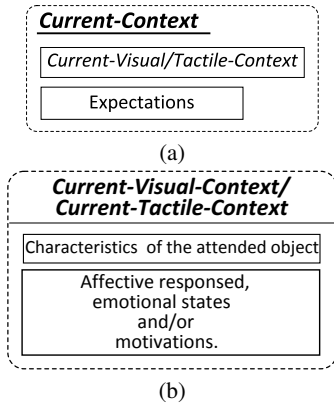


Figure 2: (a) *Current-Context* Structure; (b) *Current-Visual-Context* and *Current-Tactile-Context* structures.

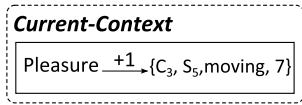


Figure 3: An example of a *Current-Context* structure.

The agent has adaptation mechanisms: the Dev E-R model

The agent has adaptation mechanisms which simulate the processes of assimilation, accommodation and equilibration described by Piaget. The way in which said mechanisms are implemented is through an extended version of the computational model of the creative process named *Engagement-Reflection* (Pérez y Pérez and Sharples, 2001; Pérez y Pérez, 2007). These mechanisms allow the agent to adapt to its world, whether by modifying its perception of the environment so that such perception is adjusted to the knowledge acquired through past experiences (i.e., adaptation by assimilation) or by modifying and creating new knowledge when this is not adjusted to the “reality” (i.e., adaptation by accommodation). These mechanisms are implemented in the *Dev E-R* model (Developmental Engagement-Reflection), which is in charge of using and constructing the knowledge of the agent (represented as sensory-motor schemas). It has two ways of achieving this: 1) automatically, through the *Engagement*

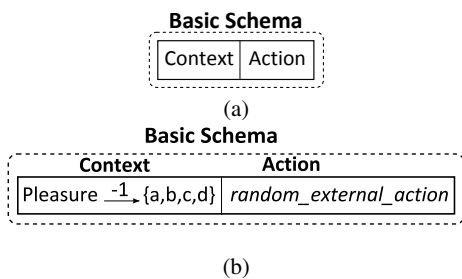


Figure 4: (a) The structure of basic schemas; (b) an example of a basic schema.

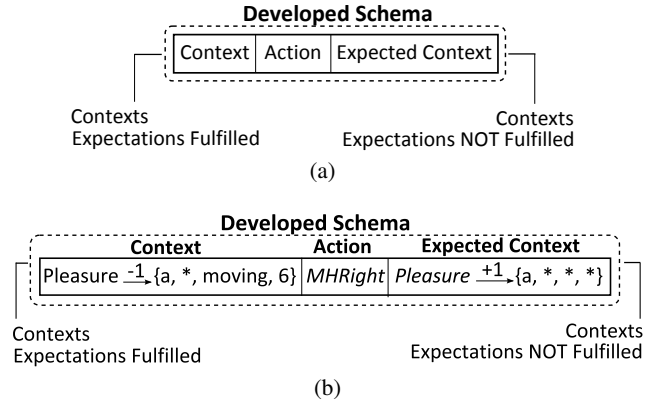


Figure 5: (a) The structure of the developed schemas; (b) an example of a developed schema.

process; and 2) analytically, through the *Reflection* process.

General Functioning The *Dev E-R* model, in *Engagement* mode, searches its memory to find schemas whose contexts represent a similar situation to the one described in the *Current-Context* (see, for instance, the case illustrated in Figure 6, wherein a 100% match exists between both structures). If during this process the agent is found to know more than one way to act given the current situation, then one of those ways is selected. The selection is performed in such a way that the developed schemas are assigned a higher probability of being chosen over the basic ones; and from the developed schemas, the one resulting in the highest number of expectations fulfilled and expected to result in the most pleasure is the one that will most likely be selected. When a schema is selected, its associated action is executed; in case the selected schema is a developed schema, then the expectations are registered in the *Current-Context* (see Figure 2). The agent senses once more its world, updating the structure of the *Current-Context*, and the cycle continues. When no schema may be matched in the memory, i.e., when the agent faces an unknown situation, then an impasse is declared. In this event, an adaptation process is required, whether by assimilation or by accommodation. These processes may be performed automatically or analytically, for instance, through analogic reasoning. However, since we are

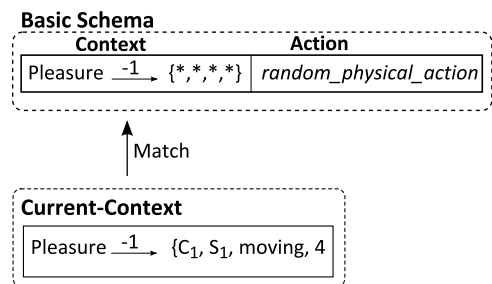


Figure 6: An example of a 100% match between a *Current-Context* and a schema available in memory.

modeling early sub-stages, the agent lacks reflexive skills to help it deal with such type of situations. Consequently, adaptation in this implementation is simulated as an automatic activity that is being performed in *Engagement* mode.

Interaction of the Agent with the World

When the agent begins its execution, it is initialized with a knowledge database that consists in a set of basic schemas, which represent reflex behaviors. From there, the agent begins interacting with the world, following the steps below:

1. The agent senses the environment through its virtual camera and its tactile sensor.
2. It creates a *Current-Context* that represents its actual “perception” of the surroundings.
3. *Dev E-R* uses the *Current-Context* structure to seek in its memory an action to perform. During this process, it is possible to perform certain modifications in the knowledge base (creation, deletion or modification of a certain schema).
4. The agent executes the action selected.
5. The agent goes back to step 1.

Steps 1 to 4 herein above are known as *perception-action cycle*.

Tests and Results

In this paper we are interested in testing the model when the agent can touch but not see the world around it (i.e. is blind), and when it can both see and touch the environment.

First test: The Agent can only touch its world

In this test the agent interacted within the living room in Figure 1a. All the objects were static, except for 5 balls that were moving as follows: 1) when the agent had its hand open and not in contact with any object, sometimes the system randomly picked any of the 5 balls and placed it in its hand (so that its touch sensor could detect it during the next cycle); 2) when the touch sensor was in contact with any object and the agent executed the action *close_hand*, then it was considered that the object had been grasped; 3) the grasped objects moved accordingly to hand movements, 4) by default, after one cycle the agent automatically opened its hand (unless when, during the current cycle, the action *close_hand* had been selected); and 4) upon the hand being opened, the object that had been grasped could remain in the same position and continue in contact with the touch sensor, or go back to its initial position (the selection above was made by the system on a random basis).

The agent was initialized with the schemas shown in Figure 7, which represent innate behaviors and tendencies described by Piaget. It was also pre-programmed with the capacity of recognizing 5 different textures on the objects it touched, labelled: “ t_1 ”, “ t_2 ”, “ t_3 ”, “ t_4 ” and “ t_5 ”. The execution began with the agent sitting in the middle of the environment with its hand open in front of it, and the 5 balls in positions that were out of its reach. We considered that the development of the agent was completed when it remained

in a state of cognitive equilibrium during the last 500 cycles. That happens when it shows to have acquired new skills that enabled it to interact with the environment by recovering and preserving the tactile objects that were pleasant for it.

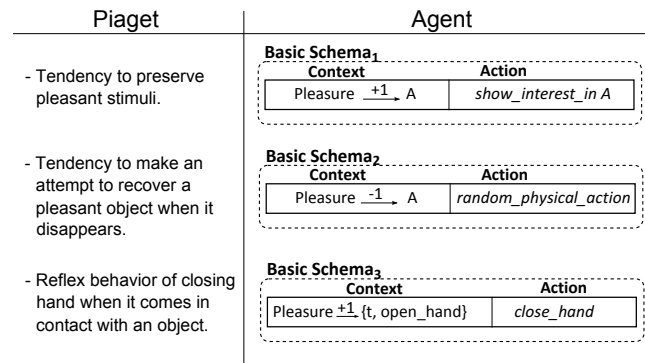


Figure 7: Basic schemas with which the agent was initialized. Letter “A” is a variable that may describe any visual object (defined as {color, size, movement, position}), or any tactile object (defined as {texture, hand_status} wherein said hand status may be *closed_hand* or *open_hand*).

In three independent executions the agent learn the same three schemas shown in Figure 8. This first schema associates the situation of having opposing affective responses caused by the same object (unpleasantness due to the loss of an element that had been grasped, and pleasure for detecting the same object on an ongoing basis, but now with the hand open), with the action of closing the hand and the expectation of recovering the affective response of pleasure resulting from grasping again the object of interest. The second schema associates the situation of having an affective response of pleasure triggered by touching any object with the open hand, with the action of closing the hand and with the expectation of having again an affective response of pleasure caused by touching the same object, but now with the closed hand. The construction of this second schema caused the agent to begin closing its hand when it came into contact with any of the objects of interest, but not as a result of a reflex behavior, (through the use of a basic schema), but rather as a result of a developed behavior having an expectation associated therewith. The third and last schema associates the situation of having an affective response of pleasure triggered by touching any object with the closed hand, with the action of closing the hand and with the expectation of maintaining the affective response of pleasure caused by holding the same object. The construction of this third schema resulted in that, from that point onwards, the agent began to maintain its hand closed when it was holding an object of its interest, which was then released when an emotional state of boredom was triggered. In other words, the agent learned to hold on to the objects of its interest.

Second test: The Agent can see and touch its world

In (Aguilar and Pérez y Pérez, 2015) we presented the results obtained when the agent could see but not touch the living room of Figure 1a. In that environment all the objects were

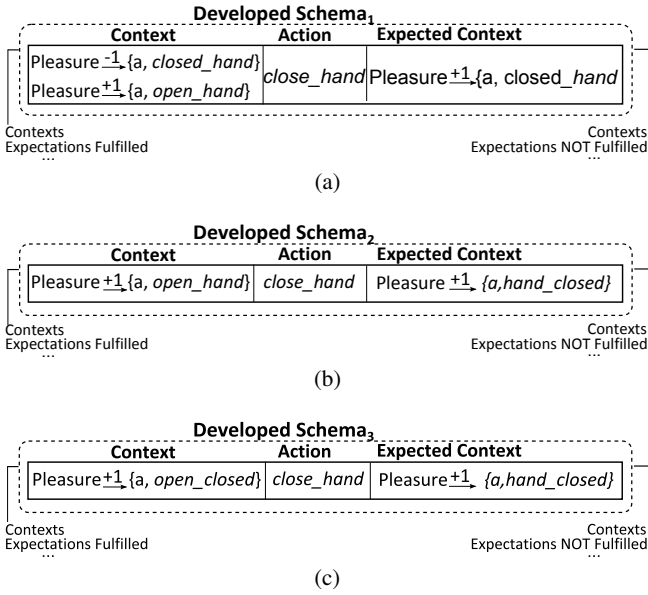


Figure 8: Schemas created when the agent was blind and it could only interact with the living room by using its sense of touch.

static, except for 5 balls of different colors, which began to move at different times and in different pre-defined directions (sometimes they rolled from left to right, and back; other times, they bounced). In that test the agent was initialized with the first two schemas of Figure 7, and at the end of its development it had learned to: 1) visually following the objects in motion, 2) centering them within its field of vision, and 3) staring at static objects in the center of its field of vision.

For this test the agent interacted again with the living room of Figure 1, but this time it was initialized with the 3 basic schemas shown in Figure 7, the schemas developed when it could only see its world, and with the schemas developed when it could only touch its world (shown in Figure 8). In other words, the agent started the execution with the knowledge of how to visually following the objects of interest, centering them in the visual field, as well as to hold on to the objects that come in contact with the hand and keeping them held until another object attracted its attention or until it became bored. The reason for the above is that, in this test we are interested in observing the set of behaviors that arise when the agent has constructed and stabilized both its visual and tactile schemas. This time, the agent started sitting in the middle of the environment, with the head looking to the front and with its hand outside the visual field. All the objects were static, except for agent’s hand, which was moving in random directions at some points in time. Later in this run, the balls began to move in the same way they behaved during the first test.

Upon starting, the first observation we found was that, when agent’s hand came into the visual field, that part of its body caught its attention and it began to follow that luminous element with head movements (using the developed

schemas that were available at initialization). It is important to note that, up to this moment, the agent sees its hand moving, not exactly because it is a part of itself (as it has not developed any knowledge structure which allows it to distinguish its own body from the rest of the objects in the environment), but because the hand catches its attention due to the color, size and movement of the hand itself, as if it were any other element present in the virtual world.

After 500 cycles, the balls began to move so as to enter into contact with the agent’s hand. From that moment, the behavior of the agent consisted in holding the objects that were in contact with its sensor, and then, releasing the items when it lost interest in them, and in following the visual elements that caught its attention. In other words, the agent continued to interact with the environment by using all the knowledge it was initialized with. After 1500 additional cycles, the agent reached a state of cognitive equilibrium, causing the schemas stored in memory to be considered stable, thus being the agent capable of: 1) representing in a same *Current-Context* both visual and tactile data; and 2) finding partial matches with the stabilized structures. These new capacities opened room for the creation of 32 new schemas, 4 per each position of the peripheral areas within the visual field. When the agent used together all its acquired knowledge, we could observe the emergence of the following behaviors: 1) visually following its hand by moving its head, 2) centering within its visual field (by moving its head) the object being held, 3) seeing within its visual field how its hand grabs and releases the object of interest, 4) seeing how its hand grabbing an object goes out of its visual field, and then recovering that image by moving its head, and 5) seeing how its hand grabbing an object goes out its visual field, and then recovering that image by moving its hand.

Discussion

In (Aguilar and Pérez y Pérez, 2014) it is presented a set of useful criteria to assess whether the behaviors generated by an agent may be considered as creative. We will discuss each of them to evaluate the results obtained in this paper.

Novelty: As presented in the previous section, when the agent could only see but not touch its world, it learned different behaviors related to visually following the objects of its interest and in centering them in its field of vision. When it could only touch but not see, the behaviors it learned were related to object grasping based only on its hand information, and when it could both see and touch it learned new skills related to grasping involving its sight and hand. All these new skills represent different behaviors from the reflex conducts defined at the beginning of the execution.

Hence, under this criteria, all behaviors it learned are considered novel.

Utility: To assess the usefulness of the behaviors developed by the agent, lets consider that it was initialized with reflex conducts wick are typical behaviors of the first sub-stage of the sensory-motor period. From there, it constructed the first schemas referred to the recovery of the interesting objects. These structures were subsequently

used as basis of knowledge, partially applying them to new situations faced by the agent, for the construction of the next schemas referred to maintaining the pleasant items. The use in conjunction of visual and tactile schemas led the agent to the acquisition of new behaviors associated with coordination of vision and touch. Hence, the knowledge structures developed by the agent are considered useful, as they allowed it to go from predefined or “innate” behaviors (typical of the first sub-stage of the sensory-motor period) to body-based behaviors (typical of the second sub-stage of the sensory-motor period) and to behaviors involving external objects (typical of the third sub-stage of the sensory-motor period).

Emergence: In the *Dev E-R* model, the learning of different behaviors depends on a number of factors, notably: 1) environmental properties, 2) physical characteristics of the agent, and 3) current knowledge. Regarding item one above, if the agent lived in a world where the task of recovering an object implied having to move the head up, then the agent would develop schemas that would represent this feature of the environment surrounding the agent. Regarding item two, when the agent was blind, it developed behaviors different from those developed when it was granted with the ability to see (using exactly the same processes of adaptation in both cases). Regarding the third point above, as discussed in the previous criteria for creativity, the behaviors developed by the agent are based on previous developed skills. Hence, we may conclude that the behaviors developed by the agent emerged as a result of the way in which the different system components interacted with each other, as the new behaviors are not pre-programmed and, furthermore, they are all context-dependant.

Motivations: One of the crucial components of the agent is that it simulates affective responses, emotional states and an intrinsic motivation of cognitive curiosity that prompt the agent to act. In particular, regarding the development of new schemas, these are created, modified or deleted as a result of: 1) an emotional state of surprise (e.g., caused by the unexpected recovery of an object of interest), or 2) a motivational cognitive curiosity that is triggered when the agent faces a conflict situation when dealing with unknown circumstances which contradict its current knowledge of the world). Hence, in this model, the emotional state of surprise and the intrinsic motivation of cognitive curiosity trigger in the agent the need to modify or construct new schemas.

Adaptation. The ability to adapt ourselves to our environment has been traditionally deemed (probably since Darwin) as a condition needed for the truly creative behavior (Runco, 2007). Additionally, as we mentioned in the introduction, adaptation and creativity are so much related to one another, that even LeoNora Cohen thinks of adaptation as the closest synonym of creativity (Runco, 2007, p.44), who describes the latter as a series of adaptive behaviors in a continuum set of seven levels of development. In Piaget’s theory, adaptation is defined in terms of the processes of assimilation and accommodation. The schemas developed

by the agent were created as a consequence of it facing unknown situations and reacting to them: 1) by assimilating the new circumstance to the previously acquired knowledge (through a process of searching the memory for a schema which represents a situation similar to the one being faced in the *Current-Context*); or 2) by accommodating the knowledge in such a way that it may adjust to the new experience (thus creating a new schema, or differentiating, generalizing or deleting an existing one). Accordingly, the construction of new structures of knowledge was carried out as a result of the simulation of a complementary process of assimilation and accommodation. In other words, they are originated as a result of the agent’s adaptation to its world.

Conclusions

Dev E-R is a computational model of early cognitive development, implemented as a creative process. It is inspired by the theories of Piaget (1952) and Cohen (1989). In a prior paper (Aguilar and Pérez y Pérez, 2015), it was explored its functionality in an artificial agent which could only see but not touch the world around it. In this paper, we had two main interests: 1) to explore what new behaviors could emerge when the agent was granted with the ability to touch but not see the world, by using exactly the same *Dev E-R* model and initial knowledge of the previous experiments, and 2) when it was able to see and touch. The results from the tests have allowed us to observe the generality of said model, in the sense that it, based on the sensorial capabilities of the agent, was able to learn new skills associated with vision, others associated with the sense of touch; and others related to both touching and seeing. These latter behaviors represent the first eye-hand coordination skills identified by Piaget, which are the base conducts needed to the developing of abilities related to goal-oriented behavior, and thus for problem solving.

Additionally, in this paper we have described features that some authors associate to creative conduct. We have employed such characteristics to suggest a criteria for the evaluation of early-creative behavior. This is an interesting problem for several reasons. Developmental agents start with few very basic knowledge-structures. Through interaction with their environment they develop new knowledge that allows them to acquire abilities that were not originally programmed. Can this behavior be considered as creative? We claim that it can as long as it fulfills the criteria of novelty, utility, emergence, adaptation and motivation. We speculate that early-creative behavior might eventually lead towards agents able of complex creative performance. If someday we are able to create a developmental system that ultimately acquires the capacity of composing music, writing poems or progressing plots, our understanding of the creative process will significantly increase. We believe that this scenario is possible. In Dev E-R we used the ER-Model that we employed in building MEXICA (Pérez y Pérez, 2007; Pérez y Pérez and Sharples, 2001), the ERI-Designer (Pérez y Pérez, Aguilar, and Negrete, 2010), and Tlahcuilo, our visual composer (Pérez y Pérez, González de Cossío, and Guerrero, 2013). So, we have the two poles of one continuum. Now, we only have to figure it out how to connect them. Probably

this will take several years. This is only the first step.

Acknowledgments

This research was sponsored by the National Council of Science and Technology in Mexico (CONACYT), project number: 181561.

References

- Aguilar, W., and Pérez y Pérez, R. 2014. Criteria for evaluating early creative behavior in computational agents. In *Proceedings of the fifth international conference on computational creativity. Ljubljana, Slovenia*.
- Aguilar, W., and Pérez y Pérez, R. 2015. Dev e-r: A computational model of early cognitive development as a creative process. *Cognitive Systems Research* 17–41.
- Amabile, T. M., and Collins, M. A. 1999. In: *R.J. Sternberg (Editor) Handbook of creativity*. Cambridge University Press, Cambridge. chapter Motivation and creativity, 197–312.
- Cohen, L. 1989. A continuum of adaptive creative behaviors. *Creativity Research Journal* 3.
- Guerin, F. 2011a. Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review* 26:209–236.
- Guerin, F. 2011b. *Sensorimotor Schema(s)*. In *Encyclopedia of the Sciences of Learning, Norbert Seel (ed.)*. Springer Science+Business Media.
- Ormrod, J. 2012. *Essentials of Educational Psychology: Big Ideas to Guide Effective Teaching*. Boston, MA: Pearson Education Inc.
- Pérez y Pérez, R.; Aguilar, A.; and Negrete, S. 2010. The eri-designer: A computer model for the arrangement of furniture. *Minds an Machines* 20(4):483–487.
- Pérez y Pérez, R., and Sharples, M. 2001. Mexica: a computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence* 13(2):119–139.
- Pérez y Pérez, R., and Sharples, M. 2004. Three computer-based models of storytelling: Brutus, minstrel and mexica. *Knowledge Based Systems Journal* 17, number 1.
- Pérez y Pérez, R.; González de Cossío, M.; and Guerrero, I. 2013. A computer model for the generation of visual compositions. In *Fourth International Conference on Computational Creativity*, 105–112.
- Pérez y Pérez, R. 2007. Employing emotions to drive plot generation in a computer-based storyteller. *Cognitive Systems Research* 8(2):89–109.
- Pérez y Pérez, R. 2015. A computer-based model for collaborative narrative generation. *Cognitive Systems Research* 36-37:36–37.
- Piaget, J. 1952. *The Origins of Intelligence in Children*. London: Routledge and Kegan Paul, 1936 (French version published in 1936, translation by Margaret Cook published 1952).
- Runco, M. A. 2007. *Creativity Theories and Themes: Research, Development, and Practice*. Academic Press in an imprint of Elsevier.
- Steels, L. 1990. Towards a theory of emergent functionality,. In *From Animals to Animats: First International Conference on Simulation of Adaptive Behaviour*, 451–461. Bradford Books (MIT Press).

Expanding and Weighting Stereotypical Properties of Human Characters for Linguistic Creativity

Khalid Alnajjar¹
alnajjar@cs.helsinki.fi

Mika Hämäläinen¹
mika.hamalainen@helsinki.fi

Hanyang Chen²
hanyang.chen@ucdconnect.ie

Hannu Toivonen¹
hannu.toivonen@cs.helsinki.fi

¹Dept. of Computer Science and HIIT, University of Helsinki, Finland

²School of Computer Science and Informatics, University College Dublin, Belfield D4, Ireland

Abstract

Many linguistic creativity applications rely heavily on knowledge of nouns and their properties. However, such knowledge sources are scarce and limited. We present a graph-based approach for expanding and weighting properties of nouns with given initial, non-weighted properties. In this paper, we focus on famous characters, either real or fictional, and categories of people, such as Actor, Hero, Child etc. In our case study, we started with an average of 11 and 25 initial properties for characters and categories, for which the method found 63 and 132 additional properties, respectively. An empirical evaluation shows that the expanded properties and weights are consistent with human judgement. The resulting knowledge base can be utilized in creation of figurative language. For instance, metaphors based on famous characters can be used in various applications including story generation, creative writing, advertising and comic generation.

Introduction

Creation and interpretation of figurative language are difficult tasks to tackle by computational means. This is due to the fact that the meaning of a figurative utterance cannot be deduced by the compositional semantic meaning of the words and syntax used, but the meaning rather lies in the pragmatics. This complicates the use of figurative language in computational creativity applications, because pragmatic meaning is open to human interpretation. Therefore when exposed to a computationally generated utterance, the reader might attribute more to it than what is actually there. That is why, when approaching figurative language from the point of view of computational creativity, it is important that the creative system knows what kind of a message is likely to be conveyed pragmatically by a certain figurative sentence.

The relations between nouns, both characters and categories, and their linked adjectival properties can be used in various creative tasks such as generating and interpreting metaphors, which are a common figurative language device. In a simplified form, by knowing that a given property is strongly related to a noun A, we can construct a nominal metaphor that indirectly conveys the meaning of another noun B having this property by stating “B is A”. However, such a metaphor doesn’t convey the meaning on the semantic level, but rather pragmatically. Therefore, by knowing

the stereotypical adjectival properties of the nouns used in the metaphor, we gain access to its pragmatic interpretation.

Consider, as an example, the sentence “Britney Spears is a cat”. Following the terminology of Richards (1936), in this metaphor “cat” is known as the *vehicle*, the noun whose property is reflected to the *tenor*, “Britney Spears”.

In order to generate such a metaphor, it is important to know the stereotypical properties the characters have. For example, to understand the previous example as equivalent to “Britney Spears is wild”, one must know that wildness is a strong property of *cats*. It is important to bear in mind that, because we are dealing with figurative language, the interpretation given as an example is not the only possible one. Therefore, it is also important to know how strongly properties are linked to nouns in order to construct metaphors that are likely to carry the intended meaning, or to reach the most plausible interpretations.

In this paper, we propose a computational method for obtaining a list of stereotypical properties for characters by expanding their given properties, using the NOC list (Veale 2016) as our starting point. This is done for famous characters (e.g. *Britney Spears*) as well as the categories these characters belong to (e.g. Singer). Our method expands a given initial set of properties provided by the NOC list with additional knowledge gathered from the internet. Our methods also weight the noun–property associations, i.e., they estimate how strongly a property is associated with a given noun.

In this paper, only the properties that are strongly linked to nouns are considered stereotypical, meaning that our approach will eradicate the properties that are in the semantic penumbra of the noun they are linked to. The word stereotype is used here in similar fashion both in the case of categories and characters, referring to the most descriptive properties, as opposed to the use of the word stereotype exclusively when describing categories of people with a prejudiced connotation.

This work is motivated by computational creativity, with the aim of providing tools to create and interpret figurative language. The method described in this paper, or its results which are publicly available, can be used as an auxiliary tool in systems such as in natural language generators to substitute literal expressions with figurative ones while still retaining the original semantic meaning. In other words, this

work provides a piece to the larger puzzle of computational creativity in the context of figurative language, both in its interpretation and generation.

This paper is structured as follows. After briefly reviewing related work, we give an overview of how the initial data is obtained for characters and categories, together with their properties and the limitations the initial data has. We then describe methods for (1) expanding the set of properties for characters and categories, (2) computing weights for the noun–property pairs, and (3) filtering out noun–property pairs with low weights. We continue by reporting on a crowd-sourced evaluation of this expansion, and then discuss the results.

Related Work

The motivation of this paper is metaphors where the tenor and vehicle are either characters or categories of people. Characters, in this case, are famous people from the real or fictional world such as *Albert Einstein* and *Batman*. While characters are all proper nouns, categories are common nouns (such as hero, scientist) that can be used to categorize people.

The work presented in this paper builds upon the foundations of a system called Thesaurus Rex (Veale 2013). In the same way as Thesaurus Rex links nouns to properties, our approach will link both characters and categories to adjectival properties.

As Searle (1958) points out, the whole semantics of a proper noun poses problems far beyond those of common nouns. From the point of view of type-token distinction (Peirce 1974), common nouns can refer to an entire type (e.g. in “dog is man’s best friend” the word dog refers to dogs in general), whereas proper nouns refer to tokens (e.g. there’s only one *Albert Einstein*).

Constructing meaningful metaphors and interpreting them requires knowledge about the tenor and vehicle, and how they interact with each other. This is done by looking into their shared properties. In the context of this paper, a property can be understood as an adjective that is stereotypically associated with a noun.

Metaphor Magnet (Veale and Li 2012) is based on the notion that stereotype expansion and property overlap are the key components in interpreting metaphors. Nouns that are used both as tenor and vehicle are first expanded with a list of stereotypical properties usually related to the nouns in question. These stereotypical properties are extracted from Google n-gram data by linguistic patterns such as “NOUN₁ is [a] NOUN₂”. After this step, the union of the properties related to a noun and its associated stereotypical properties are attributed to the noun. Properties that are in adjectival form, VERB+ings and VERB+eds, are gathered from the internet by applying a different set of linguistic patterns. However, this data was not used in its raw form to build the two knowledge bases, but rather filtered manually. The properties a metaphor conveys are understood as the intersection of the properties of the tenor and those of the vehicle. However, *Metaphor Magnet* includes a limited coverage of characters and their properties (e.g. *Albert Einstein* has only 3 properties (*educated*, *trustworthy*, and *probing*),

which makes generating and interpreting metaphors containing *Albert Einstein* infeasible.

Our motivation for this work is that metaphor generation is a knowledge hungry task. Existing metaphor processing methods for metaphor identification, interpretation, and generation rely on a huge amount of knowledge. In the field of distributional semantics, a lot of research has been done to extract word relations from large corpora, e.g., to group words automatically into semantically related categories, using methods such as Word2Vec (Mikolov et al. 2013) or LSA (Landauer and Dumais 1997). Such semantically grouped words can include nouns and adjectives if they co-occurred together in a corpus within the same context. However, these methods generalise far too much for our needs. In expanding properties, we are only interested in the adjectives that are descriptive for a given noun, i.e. not all the adjectives co-occurring with it. In addition, proper nouns rarely co-occur in text with stereotypical adjectives describing them. Therefore, such general distributional semantics approaches cannot be used in the context of this paper.

Obtaining Initial Knowledge

We employ two resources for obtaining initial knowledge of characters and categories along with their properties: Veale’s (2016) *NOC List (Non-Official Characterization list)* is used to obtain initial sets of properties related to characters; regarding properties of categories, we utilize the output of an information extraction technique described by Veale and Hao (2008a).

In the rest of this section, we explain how the NOC list is used in our approach and how the initial stereotypical properties of categories are obtained.

The Non-Official Characterization list *The NOC List (Non-Official Characterization list)* (Veale 2016) is a rich resource containing myriad information about 804 famous characters, both real and fictional. The NOC list contains so called positive and negative talking points for each character. These simply mean positive and negative adjectives that describe the character in question. In this paper, we only use the talking points and categories from the NOC list, leaving all the other information in the NOC list aside due to its irrelevance to the problem we are tackling.

Talking points of characters are used as their stereotypical properties. Positive talking points in the NOC list include words such as *funny*, *convincing*, *wise* and *powerful*, while examples of negative talking points are *bossy*, *inhuman*, *evil*, and *fat*. The talking points are not necessarily true about the characters, instead they are properties that people commonly would associate with these characters when thinking of them. In total, there are 1983 unique properties in the NOC list.

Stereotypical Properties of Categories The NOC list also contains categories of characters. They are often occupations (e.g. Actress and Scientist), but can also refer to other kinds of social groups (e.g. Child and Bully). In the

NOC list, categories do not have associated properties, because they are only used to provide the character entries with more information, i.e. they are not independent entries in the same way character entries are. On average, a character has three categories. Overall, there are 449 unique categories in the NOC list.

The initial stereotypical properties for all the categories mentioned in the NOC list were obtained by the approach described by Veale and Hao (2008b; 2008a; for the datasets, see end of this paper). The Google Search API was used to mine the web for similes with the pattern “as *ADJ* as a|an *NOUN*” with the hypothesis that an adjective *ADJ* is potentially a stereotypical property of a category *NOUN*.

Human judges were asked to annotate whether the properties were meaningful in an empty context to ensure that the properties were of a high quality and to filter out any noisy properties. Empty context here refers to the notion that the properties should make sense even when no additional cues about the context than the property and category themselves are provided.

Out of the 449 categories in the NOC list, the described approach only retrieved properties for 336 categories. This is due to various reasons that we will return to in Discussion.

Resulting Knowledge In the obtained initial knowledge base, on average, a character has 11 stereotypical properties, whereas a category has 25 stereotypical properties. The initial knowledge base has two shortcomings that the rest of this paper will tackle.

First, characters naturally have more than 11 properties. Comparing characters solely based on their properties provided in the NOC list limits the operation as some of the properties that are descriptive of them might not be stated directly in the knowledge base (e.g. *Batman* is *adventurous*).

Second, these stereotypical properties are not weighted. In other words, there is no way of telling whether a character or category is more strongly related to a given property than to another one (e.g. is *Stewie Griffin* more *evil* or more *intelligent?*).

Expanding and Weighting Properties

We expand the sets of stereotypical properties of characters and categories and weight both the initial and the newly inferred properties.

In the expansion phase, we infer new properties of a noun based on the initial knowledge base. For instance, people having the property *brave* are typically considered *adventurous*, and so are *bold*, *strong*, *agile* and *resourceful* people. As a result, *Batman* should also be seen *adventurous* as his initial knowledge states that *brave*, *bold*, *strong*, *agile* and *resourceful* are some of his stereotypical properties.

In terms of weighting these properties, our hypothesis is that the more knowledge there is to back up a given claim (such as that *Batman* is *brave*), the higher the weight should be.

Both the expansion and weighting of properties are based on viewing properties in a network of their mutual associ-

ations. We start by explaining the methodology of establishing the network, then provide the algorithm that assigns and weights new properties to existing characters based on their initial knowledge. Thereafter, we describe how weak properties are pruned.

Construction of a Property Network In order to predict more stereotypical adjectival properties for nouns, both characters and categories, we construct an undirected weighted network of properties from large corpora. The network is initialized with seed properties from the initial knowledge base.

We use Veale’s (2011) neighbouring properties dataset (see end of this paper) to obtain links between properties. Veale used the simile pattern “as *p* and * as” as in “as sweet and * as” to retrieve neighbouring properties of *sweet*. The used search engine, Google, returned matching results, e.g. “as sweet and creamy as” and “as sweet and moist as”, along with the frequencies of these phrases. Veale additionally normalized the frequencies. In total, 8644 properties are interconnected with other properties in the dataset.

Using the above-described Veale’s (2011) dataset, we construct an undirected weighted graph $G = (V, E, w)$. The set of all retrieved properties constitutes the set V of nodes. The set $E \subset V \times V$ of edges is obtained as all pairs of properties in the dataset; we consider edges undirected/symmetric. The weight $w(p_i, p_j)$ of an edge $(p_i, p_j) \in E$ is obtained from the dataset.

We use $N(p)$ to denote the set of properties adjacent to p , i.e., $N(p) = \{p_j \mid (p, p_j) \in E\}$. We use notation $P(n)$ to refer to the stereotypical properties of a given noun n in the initial knowledge base. The set of all properties known in the initial knowledge base is denoted by $P = \cup_n P(n)$.

Finding and Weighting Related Properties The main objective here is to expand the initial set of properties associated with nouns and weight them. To expand the properties of a given noun n , we iterate over all the properties in our knowledge base, P , and examine their relevance to the input noun n based on the property network constructed above.

Given a noun n , we consider one property $p \in P$ at a time. We find out which other properties are related to both n and p , supporting their mutual relationship:

$$R = P(n) \cap N(p).$$

These supporting properties are used to compute a weight $W(n, p)$ for the stereotypicality of property p for noun n :

$$W(n, p) = \sum_{r \in R} w(r, p).$$

We use uppercase $W(\cdot)$ for the resulting weights of stereotypical properties to distinguish them from the weights between properties. Recall that property pairs have weights (in the network constructed above) but noun-property pairs do not (in the initial knowledge base).

In the special case that $p \in P(n)$, i.e., our initial knowledge already indicates that p is a stereotypical property of

noun n , we give p some extra weight:

$$W(n, p) = C + \sum_{r \in R} w(r, p), \quad (1)$$

where C is a positive constant. We define it as $C = mc$, where $m = \max(w(\cdot))$ is the maximum weight of an edge, and $c = 3$ is a constant which depends on how one wants to amplify these special cases. We empirically chose $c = 3$ as a lower value would not have any noticeable effect on the weightings, and a higher value resulted in having the properties in the knowledge base be the highest.

Pruning Weak Properties For any noun n , the method described above yields a weighted list of new properties. Some of the properties may only be weakly related to the noun, however. We therefore only keep the best of them.

First, we only keep the top 20% of the properties for each noun (a character or category).

Second, the final weight $W(n, p)$ has to be greater than m , the maximum of any single weight in the property graph. This is to ensure that every new property is supported by at least two related properties, or more if their weights are smaller.

Third, once all weights are calculated for all nouns, we filter out any property that is not linked to at least two nouns. The rationale is that such properties are not helpful when comparing nouns to produce metaphors.

Results of Expansion and Weighting The described approach found and weighted new properties for 99% of characters (793 out of 804) and 82% of categories (276 out of 336). As a result of all these steps, 63 and 132 new properties were added on average to each character and category, respectively. In addition, all existing properties of nouns were given weights. The expanded and weighted noun-property pairs are publicly available (see end of this paper).

Examples We illustrate the results of property expansion with two real examples.

First, consider the person *Britney Spears*. Her initial properties in the NOC list are *tacky*, *wacky*, *sleazy*, *pretty*, *burnt-out*, *energetic* and *sultry*. Our expansion method inferred 29 additional properties for her. Added properties with the highest weights are *sexy*, *weird*, *young* and *silly*, while added properties with the lowest weights are *wild*, *vigorous*, *shiny*, *entertaining*, *skilled*, *enthusiastic* and *imaginative*. The method aims to only find stereotypical properties, so it considers all these properties for *Britney Spears*, including the ones with lower weights but still above the defined threshold. I.e., *wild* is related but less central to the stereotypical view of *Britney Spears* than what the strong properties are.

As a second example, consider one of *Britney Spears*' categories, namely *Singer*. Initially it had 21 properties, and the expansion added another 86 properties. Among all the properties, the highest weights are assigned to *expressive*, *melodic*, *artistic*, *lyrical*, *musical*, *entertaining*, *tuneful* and *gifted*. The lowest weights yet above the threshold, on the

other hand, are for *energetic*, *concise*, *capable*, *fine*, *harmonious*, *soulful* and *humorous*.

Evaluation

We next evaluate the quality of the expanded and weighted set of noun-property pairs. The evaluation is carried out using human judges, crowdsourced through the CrowdFlower¹ platform. In this section, we explain our evaluation setup and the data collected followed by the results.

Evaluation Setup

The goal of this evaluation is to validate the quality of the expanded and weighted set of properties. We do this by empirically checking if the properties and their weights correspond to what people think of them in conjunction of a given noun.

To limit the expense of the evaluation, we used a random sample of noun-property pairs. First, we randomly selected 25% of all nouns (188 characters and 73 categories, in total 261 nouns) to be evaluated. Then we selected four test properties for each noun, seeking for a diverse set of properties to evaluate. We selected one strong property (with a high weight), one "weak" property (with a low weight but still considered substantially related by the method) and two random properties that are not in the expanded set.

More specifically, we divided the noun's expanded set of properties into four equal-width bins based on their weights. We then selected one property at random from the highest bin and one from the lowest bin. For some nouns this process fails to work as intended. If the property expansion was unsuccessful, then the noun only has the initial properties, and they can all have equal weights. In these cases, both selected properties are considered strong and there is no weak property.

For every noun to be evaluated, we asked judges to rate the four selected properties on a 5-level Likert scale (Figure 1). We used five judges per noun as a trade-off between cost of evaluation and amount of data obtained. We assigned value 1 to "strongly disagree", 2 to "disagree", etc., and eventually 5 to "strongly agree". For each noun-property pair, we then used the average score from the (up to) 5 judges as the score of that pair. We compared the weights given to noun-property pairs by the proposed method to the scores given by the human judges.

The null hypothesis in our tests is that all four properties for each noun come from the same distribution, i.e., the expanded properties effectively are random and the weights do not relate to the strength of noun-property association. The values of weights and scores from judges are not directly comparable due to their different ranges; the statistics we will be using do not assume they are comparable.

Evaluators were allowed to skip nouns they did not know, e.g. a character in a movie they had not seen, by choosing "No" for the first question. In fact, the properties of a noun are shown only if the evaluator knew the noun. If they knew the noun, they were still allowed to indicate that they did not know whether the noun had a given property, as in Figure 1.

¹<https://make.crowdfLOWER.com>

Figure 1: An example of a crowdsourced questionnaire

Character: Hans Gruber

Do you know 'Hans Gruber'?

Yes
 No

Is Hans Gruber cynical?

I do not know Strongly disagree Disagree Neutral Agree Strongly Agree

Is Hans Gruber stunted?

I do not know Strongly disagree Disagree Neutral Agree Strongly Agree

Is Hans Gruber forthright?

I do not know Strongly disagree Disagree Neutral Agree Strongly Agree

Is Hans Gruber immoral?

I do not know Strongly disagree Disagree Neutral Agree Strongly Agree

The judges were limited to English-speaking countries (Australia, Canada, Ireland, New Zealand, United Kingdom and United States), and were required to have English as a language they speak in their profile. This limitation is enforced because characters in the *NOC list* are largely from Western culture and the language of all the knowledge is English. Moreover, some properties in the initial knowledge bases and the constructed property network are not commonly known even by English speakers, such as matricidal, duplicitous and loquacious.²

We did not attempt to remove likely crowdsourcing scammers as this would be difficult due to the subjectivity of how strongly nouns and properties are related. The effect of this decision is that the data is likely to contain additional noise from random entries by scammers or negligent judges.

Data

We obtained evaluations for 261 nouns (and their 4 properties) from 5 judges each, i.e., we had in total 1305 judgements of nouns.

Almost one third (31%) of these judgements were skipped evaluations where the judge indicated they did not know the character or category and thus did not score its properties. This is a relatively high rate and might be inflated due to scammers; however, the number also suggests that it was useful to allow skipping unknown nouns in order to reduce random answers and noise in the actual scores. The average number of evaluators a given noun had is 3.5.

In the following analysis, we ignore nouns that had just one or two evaluators and only consider the 199 nouns (out of 261) which had at least three evaluators. Among these 199 nouns, 127 (64%) are characters and 72 (36%) are categories. These characters are further divided to 93 real and 34 fictional characters.

The judges could also answer that they did not know if the noun had a given property. Overall, 32% of noun-property pairs received the “I do not know” answer (in all 199 considered nouns). Ignoring those two noun-property pairs that

²Based on the word’s difficulty index on Dictionary.com

only received “do not knows”, the total number of evaluated noun-property pairs for the 199 nouns is 794.

For each of the remaining 794 noun-property pairs, we have one to five Likert scores from the human judges. As mentioned above, we map the answers to values from one to five, and take the average of the answers as the score of the noun-property pair.

The inter-judge agreement on the 794 noun-property pairs by the 32 judges, using Krippendorff’s alpha measure, is 0.47.

Results

We now consider measures of how well the proposed method performed in its tasks. We first see if it can successfully identify related properties. We then consider three subtly different measures of stereotypicality of noun-property pairs and their correlation with the weights assigned by the proposed method: (1) the mean score as a direct measure of stereotypicality, (2) the standard deviation of the score as a measure of judge agreement (a proxy for stereotypicality), and (3) the number of cases when judges did not know if the noun had the property (a proxy for the inverse of stereotypicality).

Identification of Related Properties The first sub-goal of the method was to find new properties related to given nouns (without weighting them yet). We evaluated how well the method performed in this task using a two-sample permutation test for equal means. We took all new noun-property pairs that were related according to the method as one set, and contrasted them to all random noun-property pairs. The alternative hypothesis was that the mean of scores among the related properties would be higher than among the random properties. The null hypothesis of equal distributions was materialized using 10^7 random permutations of the mean scores across the two sets.

The observed difference between means was higher in the data than in any of the random permutations, yielding $p \approx 10^{-7}$. This statistically highly significant difference between the two sets indicates that the method can successfully identify new related properties using the initial set of properties and an automatically acquired network of related properties. It should be noted, however, that this is more a measure of precision than recall, i.e., the newly found properties tend to be stereotypical for the noun, but there is little information of how many truly stereotypical properties go unnoticed by the method.

Noun-Property Score Let us next take a look at the scores and weights of noun-property pairs. Noun-property pairs with higher scores are likely to be more stereotypical. For simplicity, we pool all nouns together and consider together their strong properties as one set, weak properties as one set, and random properties as one set. In this and all later experiments, where we consider weights of noun-property pairs, we include both the initial pairs and the expanded ones.

The mean scores are given in Table 1. The strong properties have a mean score of 4.13, weak properties have 3.60

Table 1: Mean and sample standard deviation of evaluation scores, and number of noun-property pairs evaluated

Categories	Strong Property			Weak Property			Random Properties		
	μ	SD	n	μ	SD	n	μ	SD	n
Real Char.	3.95	0.92	94	3.62	0.85	92	2.79	0.96	185
Fictional Char.	4.27	0.62	35	3.89	0.75	33	2.88	0.89	68
Total	4.13	0.79	206	3.60	0.85	191	2.81	0.94	397

and random properties 2.18. This indicates a clear general agreement between the human judges and the weights given by the system: the strong properties have on average 0.53 units higher scores than weak properties, and even 1.32 units higher than random properties. The scores of random properties show that judges either disagreed that a given noun has the property or found the association neutral.

For more informative statistical insight on the relation between noun-property weights and evaluation scores, we measured their correlation by simply pooling all noun-property pairs together. The random properties have no weight assigned by the system; for the test here we assumed they have zero weight. This is a very crude approach but helps us gain some insight into the correlation. The Pearson correlation coefficient is $r = 0.48$, with $p \approx 10^{-45}$. The correlation coefficient is not strong (possibly partially due to the simple approach), but the p -value indicates that the correlation is statistically highly significant and not just a random effect.

We also measured the correlation between scores and weights among the related properties only, i.e., ignoring the random properties. Pearson correlation coefficient there is $r = 0.30$ ($p \approx 10^{-9}$) suggesting that it is easier to separate random properties from related properties than strong properties from weak ones. However, the correlation between scores and weights is statistically highly significant also just among the related properties.

Standard Deviation of Scores Additional standard deviations of the scores (Table 1) can provide insight to the degree of agreement between judges. We can see that strong properties are typically more agreed on (have smaller standard deviation); however, in the case of real characters judges seem to have had slightly diverse opinions. Weak properties have higher standard deviation than strong properties, and random properties even larger, indicating less agreement and lower stereotypicality for them.

Unknown Properties In addition to the scores from human judges, we have a complementary measure of stereotypicality: how many judges knew if the noun had the property? Consider a property that has a high numerical score but was not known to be a property of the noun by many judges – such a property can not be considered very stereotypical for the noun.

Table 2 shows the percentage of noun-property pairs that were *not* evaluated by judges because they did not know whether such noun has a given property. We notice a marked

Table 2: Percentage of noun-property pairs that were rated as “do not know” the among evaluated noun-property pairs

Categories	Strong Property	Weak Property	Random Properties
Real Characters	19%	21%	54%
Fictional Characters	6%	24%	54%
Total	17%	25%	62%

increase of this number for random properties, as can be expected. Asked about a property that a character is not specifically known for, a valid answer is to say that one does not know if the character has that property. The number of unknown properties was also higher for the weak properties than for strong properties, indicating higher stereotypicality for the strong properties.

An interesting observation is that fictional characters are very well known for their strong properties (only 6% of “do not knows” vs. 19% for real characters and categories). This is probably due to the fact that fictional characters tend to have more distinctive and emphasized properties than real people; they thus seem to lend themselves better for figurative language such as metaphors.

Discussion

We have proposed and evaluated a method for expanding and weighing sets of properties of characters or categories. The empirical results, based on crowdsourcing, indicate that the method is able to identify new related properties, and to weight initial and new properties to reflect how stereotypical they are for the given noun. A number of issues were encountered during the process, however. We next discuss these issues, as well as possible applications and extensions of the proposed method.

Analysis of Problems in the Method There are three types of problems this method faced: (1) finding results matching a linguistic pattern, (2) lack of sufficient evidence to expand the knowledge base, and (3) limited initial knowledge base.

The *NOC list* contains 449 unique categories; however, for 113 categories, retrieving their properties using “as *ADJ* as a|an *NOUN*” was not successful. This is a problem of type 1 and is due to two main reasons. The first is that the category is a compound word describing another category, such as *Petty Criminal*, *Roman Gladiator* and *AI Program Villain*. The other reason is that some categories are not commonly used on the internet in the queried pattern, e.g. *Symbolologist*, *Lexicographer* and *Hyperchondriac*.

The approach for expanding properties was unable to expand the properties of 11 characters and 60 categories. Regarding characters, the expansion typically failed because there were few links to the character’s initial properties. An example of such a case is *Tiger Woods* – a golf player – who has five properties that are not available in the constructed property network, namely *philandering*, *field-topping*, *world-beating*, *highly-paid*, *long-driving*. This is also a type 1 problem as the links in the network were obtained from the simile pattern “as p and * as”. Additionally,

his remaining four properties, *promiscuous*, *unfaithful*, *athletic*, and *masterful*, are not strongly enough related to each other to infer new links with a weight higher than our defined threshold m , a problem of type 2.

The above two factors also affect the expansion of categories' properties. In addition, some categories have a very limited number of properties retrieved for them in the initial knowledge base, i.e. a problem of type 3. For instance, the categories Linguist and Frontiersman have only one stereotypical property linked to them which is *fluent* and *adventurous*, respectively.

Hence, this approach is expected to work when nouns have a sufficient number of properties (at least ~ 5) that are related to each other and exist in the constructed network. In our case study, this seems to have been the case for most characters and categories. In case where this is not feasible, the pruning conditions can be made lenient to result in higher coverage (e.g. selecting top 50% instead of 20% or reducing the threshold to $m/2$). This can add noise, however, and it requires more experimentation to find out what the exact effects would be.

Applications A direct use case for the proposed approach is in situations where a wider range of properties are required to perform a creative task but only relatively small number of properties are available at hand. For instance, Meta4meaning (Xiao et al. 2016) – a creative corpus-based system for interpreting metaphors – has shown good results of how a creative system can produce interpretations similar to humans. Nevertheless, the system was unable to interpret some metaphors (e.g. “the woman is a cat”) due to reasons including that a given noun (*woman*) was not associated with a desired property (*wild*).

Our approach for expanding the list of properties of a noun can be employed in such a scenario. The noun *woman* is not on the NOC list, so we use as examples two specific women instead. Consider the expression “Britney Spears is a cat” and its possible metaphorical meanings. One approach to find potential such meanings is to look at the shared properties of *Britney Spears* and of *cat* (and then pick some of those based on various criteria (Xiao et al. 2016)).

The intersection between the expanded properties of *Britney Spears* and *cat* include properties such as *energetic*, *spry*, *vigorous*, *wild*, etc., all possible interpretations of the metaphoric expression.

The shared properties between *Hillary Clinton* and *cat*, in turn, include words such as *smart*, *independent*, *intelligent*, etc, which are possible metaphorical meanings of the expression “Hillary Clinton is a cat”.

Veale (2016) outlines how the NOC list can be used in metaphor generation in the case of characters. In his paper, metaphors are represented as concept pairs that are constructed by using multiple overlapping properties, such as “Hillary Clinton could be Princess Leia: driven yet bossy”. His approach provides multiple recommendations for possible metaphors. For a tenor, such as *Tony Stark* (Ironman) and a desired property, such as *rich*, the system outputs possible characters to be used in a metaphor, for example “Tony

Stark is Bruce Wayne”. The extended weights from our approach can be used to generate these kinds of metaphors in a richer way, since our system provides more knowledge about the stereotypical properties and their weights. This could directly be tested out, for example, with the metaphor generation algorithm proposed by Veale and Li (2012).

The proposed approach is valuable also in creative systems requiring an input from the user, whether they are co-creative systems or not. An example of such a case is generating creative slogans (using BrainSup (Özbal, Pighin, and Strapparava 2013) for instance). Users of these systems are expected to specify the target words such as the brand name and its essential properties to highlight. Our property expansion approach can be utilized in this context to expand the initial set of properties input by the user and weight them to improve the slogans generated.

Properties of Characters in the Context of a Category

There are various possible ways to extend the proposed method; we here discuss one interesting avenue that could easily be implemented on top of the current method.

Sometimes an important aspect in metaphors is to know how strong a given property is to a noun when examined from the point of view of a given domain or category. For instance, the weights of the stereotypical property *arrogant* of *Tony Stark* when seen as Hero should be different than when seen as Billionaire.

We hypothesize that such cases can probably be handled by a simple generalization to the definitions of this paper. Assume that n is a character, c a category and p a property. The supporting set of properties is then simply constrained to those that are also properties of category c :

$$R = P(n) \cap N(p) \cap P(c).$$

Equation 1 can then be applied as before. Validation of this technique is a topic for future work.

Conclusions and Future Work

In this paper, we have presented a way of expanding a given initial set of adjectival properties for nouns to cover a wider range of their stereotypical properties, and of weighing the properties. We have successfully applied this approach both in the case of common nouns (categories) and proper nouns (characters). We also conducted an evaluation with human judges to verify the quality of the results obtained by our proposed method.

Based on the new knowledge we constructed about characters and their linked properties, future research can be conducted on computational linguistic creativity, such as metaphor interpretation and generation. An evaluation of metaphors generated using the properties and weights produced by the proposed method would also give additional insight into the quality and usefulness of the results of this paper. Such an evaluation could also inform us about whether metaphors including proper nouns are seen by people in a similar fashion as metaphors only consisting of common nouns. Based on our results, fictional characters look especially promising for metaphors since their stereotypical properties tend to be well known (cf. Table 2).

We have only discussed the expansion of a list of adjectival properties for nouns in this paper. However, the expansion of the nominal components, i.e. categories and characters has been left aside. Given that the origin of the nouns, namely the NOC list, is hand crafted and currently the only way of expanding it is by a laborious manual process, it would be interesting to see in the future if our approach can be used to expand the nominal knowledge as well, for instance, by altering the linguistic patterns.

A possible future direction for this research is to expand it to multiple languages. From a theoretical point of view, there is nothing heavily language dependent that would hinder the adaptation of this method to different languages. Since our approach deals with stereotypes which are known to be socially constructed and thus culturally dependent, this method, in the context of multiple languages, could shed more light on stereotypical beliefs in different cultures.

Datasets

The datasets used as input and produced as output by the methods described in this paper are publicly available at <https://github.com/prosecconetwork/ThesaurusRex/>.

Acknowledgements

This work has been supported by the Academy of Finland under grant 276897 (CLiC) and by the European Commission coordination action PROSECCO (Promoting the Scientific Exploration of Computational Creativity; PROSECCO-network.EU). The authors would like to thank Tony Veale for providing the data sets and inspiration for this work.

References

- Landauer, T. K., and Dumais, S. T. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review* 104(2):211–240.
- Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.
- Özbal, G.; Pighin, D.; and Strapparava, C. 2013. Brain-Sup: Brainstorming Support for Creative Sentence Generation. 1446–1455. Sofia, Bulgaria: Association for Computational Linguistics.
- Peirce, C. S. 1974. *Collected papers of Charles Sanders Peirce*, volume 2. Harvard University Press.
- Richards, I. A. 1936. *The Philosophy of Rhetoric*. London: Oxford University Press.
- Searle, J. R. 1958. Ii.proper names. *Mind* 67(266):166–173.
- Veale, T., and Hao, Y. 2008a. Enriching wordnet with folk knowledge and stereotypes. In *In Proceedings of the 4th Global WordNet Conference*, 453–461. Szeged, Hungary: Juhász Press Ltd.
- Veale, T., and Hao, Y. 2008b. Talking points in metaphor: A concise usage-based representation for figurative processing. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, 308–312. Amsterdam, The Netherlands: IOS Press.
- Veale, T., and Li, G. 2012. Specifying viewpoint and information need with affective metaphors: A system demonstration of the metaphor magnet web app/service. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, 7–12. Jeju Island, Korea: Association for Computational Linguistics.
- Veale, T. 2011. Creative language retrieval: A robust hybrid of information retrieval and linguistic creativity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 278–287. Stroudsburg, PA, USA: Association for Computational Linguistics.
- Veale, T. 2013. A service-oriented architecture for computational creativity. *Journal of Computing Science and Engineering* 7(3):159–167.
- Veale, T. 2016. Round up the usual suspects: Knowledge-based metaphor generation. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, 34–41. San Diego, California: Association for Computational Linguistics.
- Xiao, P.; Alnajjar, K.; Granroth-Wilding, M.; Agres, K.; and Toivonen, H. 2016. Meta4meaning: Automatic metaphor interpretation using corpus-derived word associations. In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016)*. Paris, France: Sony CSL.

Creative Flavor Pairing: Using RDC Metric to Generate and Assess Ingredients Combinations

Álvaro Amorim, Luís Fabrício W. Góes, Alysson Ribeiro da Silva and Celso França

Programa de Pós Graduação em Engenharia Elétrica

Pontifícia Universidade Católica de Minas Gerais

Minas Gerais, Brasil 30535-901

{alvaro.amorim,lfwgoes,alysson.ribeiro,celso.franca}@pucminas.br

Abstract

Creating culinary recipes is one of the most creative human activities. It requires combining ingredients, performing the recipe steps, creating specific diets, and others tasks. In addition to it, the existence of publicly available repositories of recipes, as well as scientific advances in areas such as Food Chemistry and Neuro-Gastronomy, encourage the generation of *new* and *pleasurable* recipes from algorithms. Although the number of ingredients allows the generation of a huge number of recipes ($\sim 10^{15}$), only a small fraction of this potential is exploited ($\sim 10^6$). This paper proposes, implements and analyzes a system called *Creative Flavor Pairing* which acts cooperatively with different profiles of cooks assuming the responsibility of suggesting food ingredients that can lead to creative recipes. These ingredient combinations are generated by a genetic algorithm using the Regent-Dependent Creativity (RDC) metric as a fitness function. Our experimental results showed that the RDC metric can be applied to the culinary field as our system was able to suggest creative ingredient combinations that match the most popular ones currently published in the largest cooking social networks.

Introduction

Computational Creativity is the term used to describe a research sub-field in Artificial Intelligence, which studies how to build software that demonstrate creative behaviors (Colton 2012; Besold, Schorlemmer, and Smaill 2015). In more practical terms, this area investigates how to create algorithms to generate results that would be considered as creative as if they were produced by humans.

Unlike common approaches where a program is a mere tool to reinforce human creativity, in the Computational Creativity research field, there is an effort to create software that is independently creative, either to act as a collaborator with people or to act autonomously as an artist, musician, writer, draftsman, engineer or scientist (Besold, Schorlemmer, and Smaill 2015; Boden 2009). Although the term *creative artifact* is often used primarily for artistic artifacts such as music, painting or poetry, it also includes innovative scientific theories, mathematical concepts, and science and engineering projects.

Recently, the culinary domain has attracted attention not only from researchers in areas like Food Chemistry,

Psychophysics or Neuro-Gastronomy, but also several researchers in Computational Creativity (Morris et al. 2012; Veeramachaneni et al. 2010; Sawyer 2012). The works of (Varshney, Wang, and Varshney 2016) and (Pinel, Varshney, and Bhattacharjya 2015) have become popular because they propose both a form of recipe generation and a metric of creativity that combine *Bayesian Surprise* and a human flavor perception model.

In addition to it, there is currently publicly available repository of recipes, food composition and the principles of tasty dishes (Ahn et al. 2011). On the other hand, the relatively small number of recipes in use ($\sim 10^6$) compared to the huge number of potential recipes ($\sim 10^{15}$) together with the frequent recurrence of particular combinations in several regional cuisines indicate that we are exploring but a small fraction of the possible combinations (Ahn et al. 2011).

A hypothesis, which in the last decade has received attention among some chefs and food scientists, claims that ingredients that share flavor compounds are more likely to generate a tasty recipe (Ahn et al. 2011). This *Food Pairing Hypothesis* has been used to find new combinations of ingredients and led, for example, some contemporary restaurants to combine white chocolate and caviar because they share certain flavors, or chocolate and cheese that share at least 73 flavor compounds.

Bearing in mind that exploring all aspects of creativity involved in generating a culinary recipe is a very hard problem, in this paper we propose, implement and analyze a creative computing system called *Creative Flavor Pairing* to act cooperatively with different profiles of cooks assuming the responsibility of selecting foods that can generate surprising and tasty ingredients combinations.

Creative Flavor Pairing uses a genetic algorithm to generate combinations of food ingredients guided by the RDC metric proposed in (França et al. 2016). This general purpose metric has been used in other domains such games and fashion, and in this work our main contribution is to verify its applicability also in the field of culinary. In order to test our proposed system, we present: i) one case study on *Allrecipes*¹ social network; ii) a quantitative evaluation of human-made recipes and recipes created by *Creative Flavor*

¹Allrecipes is the largest social network focused on food. It can be accessed at: <http://allrecipes.com/>

Pairing; and iii) a recipe made by a chef using a combination suggested by the system.

This paper is organized as follows. In the Food Pairing Hypothesis section, the *Food Pairing Hypothesis* is explained by details. Next, in the RDC Metric section is presented the method used to assess the creativity of an artifact and its model in the culinary domain. Furthermore, the Related Work section presents a brief history of creative assessment methods and its application in the culinary domain. Next, in The Creative Flavor Pairing section, our proposed creativity system is presented. The Experimental Method section outlines the experimental setup while the Experimental Results section presents and analyses the experimental results. A discussion and conclusion are presented at the Conclusions section.

Food Pairing Hypothesis

In an attempt to combine salty foods with chocolate, chef Heston Blumenthal (Blumenthal 2008) found that the combination of white chocolate and caviar resulted in a very pleasant taste. In order to identify why this and other combinations had a good result, he made analyses on the foods involved and identified that foods that had common flavor compounds, when combined, produced pleasant and tasty results. This hypothesis became known as *Food Pairing Hypothesis*.

To validate the *Food Pairing Hypothesis*, extensive work was done on (Ahn et al. 2011), where recipes from various regions of the world were evaluated to determine whether or not the ingredients involved shared flavor compounds as determined by evaluated hypothesis.

The results obtained in (Ahn et al. 2011) have shown that North American and Western European recipes follow the *Food Pairing Hypothesis*, and that this is due to some key ingredients commonly used in recipes. For the South European and East Asian regions, a rule contrary to the hypothesis was observed. In these regions, the recipes seem to avoid ingredients that share flavor compounds. This work considers the recipes of the regions where the *Food Pairing Hypothesis* was verified.

RDC Metric

Figure 1 shows the overview of the RDC metric proposed in (França et al. 2016) to evaluate the creativity of artifacts from different domains. The metric combines *Bayesian Surprise* and *Synergy* to measure novelty and value respectively.

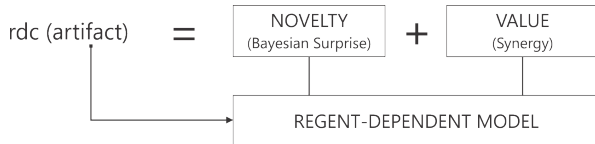


Figure 1: RDC Metric Overview.

The *Regent-Dependent Model* (RD Model) is used to assign a set of characteristics on which the creativity of an artifact is assessed in a given context. In this model, the

characteristics of an artifact are represented by *pairs* p_i associated with a numerical value v_i :

$$p_i(\text{regent}, \text{dependent}) : v_i$$

Where *regent* indicates a characteristic of an artifact and *dependent* defines the state of that characteristic. The value v_i expresses the intensity of *pair* p_i when describing an artifact in a given context.

RD Model in the Culinary Domain

In the culinary domain, there is no doubt that our flavour experience is mainly made up from sensations of taste, touch (texture) and smell(aroma) (Page and Dornenburg 2008). A recipe can be characterized by its list of ingredients. Each ingredient, in turn, is modeled on the RD Model in which the taste, texture and aroma are the *regents* and the *dependents* are the various tastes, textures and aromas that a food has, as shown in Table 1.

regents	dependents
taste	..., sweet, salty, bitter, sour, umami, ...
texture	..., crispy, soft, juicy, consistent, ...
aroma	..., toasted, citric, cheesy, ...

Table 1: *Regents* and some instances of *dependents* used to describe ingredients present in a recipe.

As an example, Table 2 lists eight *Regent-Dependent pairs* used to describe the taste, texture and aroma of a strawberry (Burdock 2016).

taste	$p_1(\text{taste, sweet}): 0.49$ $p_2(\text{taste, sour}): 0.51$
texture	$p_3(\text{texture, soft}): 0.38$ $p_4(\text{texture, juicy}): 0.62$
aroma	$p_5(\text{aroma, fruit}): 0.56$ $p_6(\text{aroma, toasted}): 0.04$ $p_7(\text{aroma, cheesy}): 0.25$ $p_8(\text{aroma, citric}): 0.05$

Table 2: *Pairs Regent-Dependent* used to describe a strawberry.

Novelty Metric in the Culinary Domain

Although still under discussion, some authors have adopted surprise as an emotional response that acts as a novelty detector (Grace and Maher 2016; Varshney et al. 2013; Wiggins 2006). In this article, we also use the well-known Bayesian Surprise (Baldi and Itti 2010) as a metric of novelty.

Considering an artifact as an event, the amount of information calculated by the Bayesian Surprise can be interpreted as the novelty $n(R)$ of an artifact R , as defined in Equation 1 (Baldi and Itti 2010).

$$n(R) = \delta[P(M); P(M|R)] \quad (1)$$

Where the probability distribution $P(M)$ represents the degree of belief of an observer in model M and $P(M|R)$ reflects the knowledge after a new artifact R has occurred and been inserted in the model M . Therefore, Equation 1 assumes that the novelty contained in an artifact R can be measured by considering the difference δ between probability distributions that describe how the observer's worldview changed from the occurrence of R .

The beliefs of an observer $P(M)$ define a context represented by a dataset of known artifacts. In the *Creative Flavor Pairing*, the dataset is a set of ingredients combinations represented in the RD model and arranged in rows and columns. Lines are the combinations and columns are the pairs used to describe them.

	p_1	p_2	...	p_j	...	p_{n-1}	p_n
a_1	v_{11}	v_{12}	...	v_{1j}	...	$v_{1(n-1)}$	v_{1n}
a_2	v_{21}	v_{22}	...	v_{2j}	...	$v_{2(n-1)}$	v_{2n}
...
a_i	v_{i1}	v_{i2}	...	v_{ij}	...	$v_{i(n-1)}$	v_{in}
...
a_{m-1}	$v_{(m-1)1}$	$v_{(m-1)2}$...	$v_{(m-1)j}$...	$v_{(m-1)(n-1)}$	$v_{(m-1)n}$
a_m	v_{m1}	v_{m2}	...	v_{mj}	...	$v_{m(n-1)}$	v_{mn}

Table 3: Dataset of known ingredients combinations. The dataset organizes in rows and columns the combinations represented in the RD model.

Table 3 shows a dataset containing m recipes described by n pairs. Thus a recipe R_i is placed on line i and a pair p_j , used to represent a characteristic of R_i , is mapped to a column j . The value v_{ij} representing the intensity of p_j in R_i , is copied to the position ij of the dataset.

Thus, the novelty of an recipe is equal to the sum of the novelties of the pairs used to describe it, as shown in Equation 2.

$$n(R) = \sum_{p_i \in R} n(p_i) \quad (2)$$

Mathematically, the function used to compute the novelty $n(p_i)$ of a pair p_i used to describe a particular recipe R , is of the form as shown in Equation 3 (Baldi and Itti 2010), where σ and \bar{m} are respectively the variance and mean of the pairs in the recipe dataset and v_i is the value associated with p_i .

$$n(p_i) = \frac{1}{2\sigma_i^2} \left[\sigma_i^2 + (v_i - \bar{m}_i)^2 \right] \quad (3)$$

Since there is no limit on how new an artifact can be, Equation 4 normalizes novelty in the interval $[0, 1]$ by an exponential normalization.

$$f[n(A), \lambda] = 1 - e^{-\lambda n(A)} \quad (4)$$

Where λ is a positive real number known as *smoothing factor*. The greater the *smoothing factor* λ , the more expressive are small novelties.

Value Metric in the Culinary Domain

There are plenty of information publicly available (Grace et al. 2014) that describes recipe and the elements that constitute them like in *fooDB*². In particular, it is also available how these elements interact and what interactions are most valued in a given context, which is key to compose a valuable artifact.

In culinary, a recipe is valued when the combination of its ingredients produces a tasty recipe. The *Food Pairing Hypothesis* states that ingredients will work well together in a dish if they share similar flavors. These facts give evidence that the relationship between the ingredients of a recipe can be used as a measure of value.

The approximation of how a recipe is valued in a context is carried out in a *synergiset*. A *synergiset* is a way of specifying which ingredients of an recipe are synergistic. Synergistic ingredients are those described by synergistic pairs. And these, in turn, represent characteristics (taste, texture and aroma) that when they occur together are responsible for the value of a recipe.

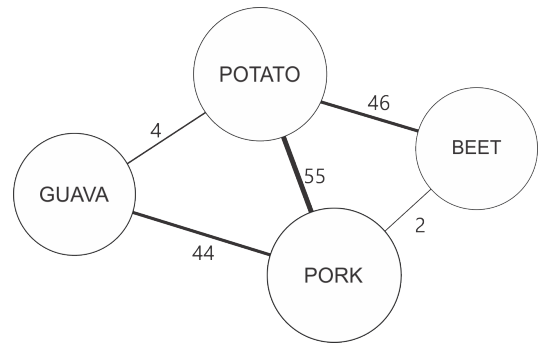


Figure 2: An example of synergiset.

²fooDB is the world's largest and most comprehensive resource on food constituents, chemistry and biology available at <http://foodb.ca/>

The *synergiset* structure is a graph $S(V, E)$. Figure 2 illustrates a part of the *synergiset* composed of 4 vertices representing the ingredients: pork, potato, beet and guava. The complete *synergiset* consists of 1529 vertices representing all ingredients currently available in our database to compose a recipe.

There is an edge between ingredients when they are described by synergistic *pairs*. According to the *Food Pairing Hypothesis*, a pair p_i is synergistic to a pair p_j when they have the same *regent* and same *dependent*, that is, when two ingredients share the same flavor compound. The weight w_{ij} of the edge between the ingredient i_i and i_j indicates the level of synergy between two ingredients. It is equal to the number of synergistic *pairs* between that two ingredients.

A graph G_R representing the ingredients of a recipe R is defined by the subgraph of S formed by the vertices representing the ingredients in R . From the graph G_R , the value $v(R)$ of a recipe is calculated, as defined in (França et al. 2016) by Equation 5.

$$v(R) = \frac{1}{2}kc(G_R) + \frac{1}{2}\rho(G_R) \quad (5)$$

Where $kc(G_R)$ is the Krackhardt's connectedness of G_R (Krackhardt 1994) and $\rho(G_R)$ is the density of G_R (Matta et al. 2016). The first term of the Equation 5 measures the associativity among the ingredients of a recipe. If all ingredients are synergistic, then *Krackhardt's connectedness* of G_a is maximum. If all ingredients are isolated in the recipe, then $kc(G_a)$ is minimum. The second term measures the strength of the connection among the ingredients.

The Regent-Dependent Creativity (RDC) Metric

The RDC metric for the creativity evaluation of the ingredient combination of a recipe R is defined in Equation 6 as the sum of the novelty $n(R)$ and of the value $v(R)$ together with a penalty function. This penalty is necessary to avoid that new and low-value recipe (different but useless) or valuable recipes with low novelty (useful, but already known) are considered creative.

$$rdc(R) = n(R) + v(R) - p[n(R), v(R)] \quad (6)$$

$$p[n(R), v(R)] = s(1 - e^{-kd}) \quad (7)$$

where:

- s : is the sum of $n(R)$ and $v(R)$.
- d : is the absolute difference between $n(R)$ and $v(R)$.

Equation 7 penalizes the creativity of an artifact depending on the difference among its novelty and its value. The greater the difference between novelty and value of an artifact, its creativity is more penalized. The penalty is more intense as the variable k is higher, however, no artifacts are penalized more than the sum of its novelty and its value. Therefore creativity is in the range $[0,2]$.

Related Work

In Computational Creativity, an assessment method allows machines to generate and evaluate creative artifacts (Boden 2004). It was realised the importance of a consensual way to evaluate creative thinking since (Amabile 1982; Partridge 1985; Ortony and Partridge 1987) allowed machines to perform in a similar way as human beings. At that time, most of the assessment methods were based on psychology, describing it as a mental process that involves surprise, expectancy and luck (Stiensmeier-Pelster, Martini, and Reizenzein 1995). During the development of a consensual method to evaluate creativity, the surprise started to be used to create artificial agents as presented by (Macedo and Cardoso 2001; Macedo, Reizenzein, and Cardoso 2004), thus allowing to model novelty into its behavior or design. Assessment methods evolved as a set of mathematical equations that describes creative artifacts as a combination of surprise, novelty and value as shown by (Grace et al. 2014). To this date, researchers tend to agree that an artifact has to be new and valuable on a particular domain to be considered creative (França et al. 2016; Goes et al. 2016; Boden 2015; Colton et al. 2015; van der Velde et al. 2015)

Another popular approach is the use of human computation, in which artifacts' creativity is assessed by human judges (Lamb, Brown, and Clarke 2015). Human computation to assess creativity can be found in fields such as fashion, focusing on the creation of fashion styles and designs (Cheng and Liu 2008), the creation of full playable digital games (Cook and Colton 2015), and even on the development of autonomous agents (Goel and Rugaber 2015). Those systems can also utilize collaborative methods as presented by (Joyner et al. 2015), where the most creative artifact is the product of many others. Human computation can be a value estimator as pointed by (Jordanous, Allington, and Dueck 2015), where the authors show how humans evaluate music composed by others. It is important to note that the quality of a judge evaluation is highly tied to the expertise of the judge, thus this kind of system is more suitable to assist humans in tasks than to be used in automated processes. Also, the quality of the evaluation by human systems rely in sociocultural aspects as pointed and explored by (Jordanous 2012; Jordanous 2013; Jordanous, Allington, and Dueck 2015).

Theories about creative thinking vary from the incubation theory to the most recent one called the honing theory (Gabora 2010). When transformed into computational systems, those use search heuristics, evolutionary computation and AI learning systems such as artificial neural networks or knowledge-based systems. With a consensual assessment method to evaluate creative artifacts, the research conducted by (Kim and Cho 2000; Kowaliw, Dorin, and McCormack 2012; Tomasic, Znidarsic, and Papa 2014; Cook and Colton 2015) has been using genetic algorithms driven by context bounded objective functions that consider surprise or value to evaluate creative artifacts in fields such as fashion, slogan creation and artistic painting.

In culinary domain, evolutionary algorithms seems suitable to generate creative artifacts. For instance, IBM (Varshney et al. 2013) has been successful in generating creative

food recipes by combining a suitable domain knowledge database (i.e. food ingredients, existing recipes etc.), genetic algorithms, novelty and pleasantness assessment metrics. Computational creativity in the culinary field was also discussed in (Morris et al. 2012), where the authors focus only on soups rather than general recipes. A genetic algorithm was implemented to generate soup recipes in which multilayer perceptron neural networks were used as fitness function. These neural networks were trained over user reviews from *Allrecipes* social network. In (Veeramachaneni et al. 2010) is demonstrated a particle multi-objective particle swarm optimization to generate a combination of ingredients capable of pleasing human evaluators. The objective function of the algorithm, in turn, was designed by genetic programming using the evaluation of flavors of different combinations of ingredients.

As an alternative to the evolutionary approach, (Varshney, Wang, and Varshney 2016) updates the proposal made in previous works (Pinel, Varshney, and Bhattacharjya 2015; Varshney et al. 2013). As well as previous studies, the new proposal maintains Bayesian Surprise and a regression model for assessing respectively, novelty and value (pleasantness of the flavors) of the recipes generated by the system. The main contribution is in how the recipes are generated, which is based on rules of association of ingredients considering factors such as: co-occurrence in recipes, shared flavor compounds, being from same region of world, and being grown in the same season of year.

The main contribution of this research in relation to those aforementioned is the employment of *RDC*, a domain independent creativity metric, in the culinary field. The *RDC* measures creativity through the *novelty* and *value* of artifacts. The *novelty* is measured by the *Bayesian Surprise*, also used in (Varshney et al. 2013; Varshney, Wang, and Varshney 2016), and the *value* is calculated through *Synergy*, presented in (França et al. 2016) for other application domains. To characterize the *value* of the artifacts in the culinary domain, the *Food Pairing Hypothesis* verified in (Ahn et al. 2011) was used. This hypothesis was the base to create the synergy graph, which characterizes ingredients that if combined, add value to a food ingredient combination.

The Creative Flavor Pairing

Figure 3 shows the overview of *Creative Flavor Pairing*. The *Pairing Builder* is a genetic algorithm (GA) that performs the recipe generation using the *RDC* metric as a fitness function, as established in Equation 6. The *dataset* and *synergysset* define respectively, the context for the calculation of novelty and value.

In each generation, a population of 30 candidates ingredient combinations is submitted to the operators such as: selection, crossover and mutation. The crossover operator implements *Partially Matched Crossover* (Sivanandam 2008) (crossover probability $p_C = 0.7$) and the mutation operator exchanges one ingredient from one combination for another one from the universe of the 1,530 ingredients currently available (mutation probability $p_M = 0.05$). When evolution is no longer significant, *Pairing Builder* returns

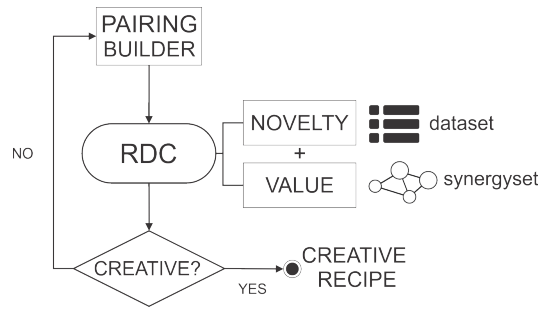


Figure 3: Overview of the Creative Flavor Pairing system.

the most creative recipe found. The implementation uses the *RDC API*³ provide in (França et al. 2016).

Experimental Method

The experimental analysis was conducted in two stages. In the first stage, a case study was carried out in the *Allrecipes* culinary social network to identify the relationship between the *RDC* metric for combinations of ingredients and the popularity of recipes created by humans. In the second stage, the novelty, value and creativity (*RDC*) of the combinations generated by *Creative Flavor Pairing* were compared to the combinations of human-made recipes ingredients published in *Allrecipes*.

The *Food2Fork API*⁴ was used in the case study in order to retrieved recipes with the lowest and highest *social ranking* (*SR*), a real number which aggregates criteria such as the number of users who reproduced and reviewed a recipe, rating (0 to 5 stars) given by the community and the number of likes and shares, from *AllRecipes*. Furthermore, the *RDC* of the ingredient combinations from the collected recipes was calculated to be clustered, through *K-means algorithm* (Witten, Frank, and Hall 2011), based on novelty, value and *RDC* of the recipe ingredient combinations, to verify if the social ranking and *RDC* have the same behavior.

In the second stage, we used the repository presented by (Ahn et al. 2011), that contains 41,524 human-made recipes retrieved from *Allrecipes*. The novelty, value and *RDC* from all the ingredients combinations in the repository were compared with the same amount of combinations that were generated by *Creative Flavor Pairing*.

Experimental Results

The Figure 4 shows the *SR* normalized in the range [0,2] and the *RDC* of the least popular and the most popular ingredient combinations from recipes obtained by *Food2Fork API*.

All least popular combinations have *SR* equal to 0.698. Considering that their *RDC* is in the range between 0.79 and

³*RDC API* is published in Github and can be accessed at: <https://github.com/CreaPar/RDC-API/>

⁴*Food2Fork* offers an API that exposes some ingredient search functionality across multiple online recipe databases. *Food2Fork* documentation can be found at <http://food2fork.com/about/api>

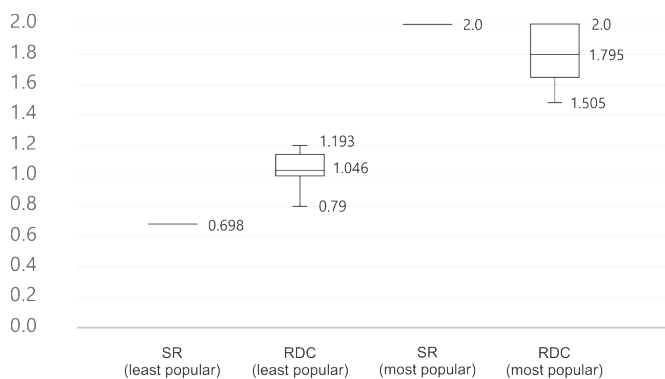


Figure 4: Comparison of SR and RDC of the least popular combinations and the most popular combinations retrieved by the Food2Fork API.

1.193 with the mode being 1.046, it indicates that the less popular combinations are also the less creative ones.

On the other hand, all most popular combinations have a *SR* equal to 2.0. And their RDC is between 1,505 and 2,0 with the mode being 1,795. That is, the most popular combinations are also the most creative.

	novelty	value	RDC
Cluster 0	0.592	0.882	1.089
Cluster 1	0.926	0.942	1.799

Table 4: Novelty, value and RDC of centroids of Cluster 0 and Cluster 1.

Table 4 shows the novelty, value and RDC of the centroids resulting from the clustered combinations. The *Cluster 0* has its centroid with RDC equals to 1.089, while the *Cluster 1* has its centroid with RDC equals to 1.799. These clusters concentrate ingredients combinations of low and high creativity, respectively. All combination of *Cluster 0* has *SR* equal to 0.698 while all those of *Cluster 1* has *SR* equal to 1.799. This fact confirms that the most creative recipes are those that are most successful among *Allrecipes* users.

As results of the second stage of experiments, Figure 5 shows that the average of novelty, value and RDC of human-made ingredient combinations and combinations created by our system. There is a relatively small number of recipes in use, ($\sim 10^6$), compared to the huge number of potential recipes, ($\sim 10^{15}$), together with the frequent recurrence of particular combinations in several regional cuisines. For this reason, the genetic algorithm in our system, in less than 30 generations, was able to find combinations with considerably greater novelties than the combinations created by humans, as shown in Figure 5(a).

In addition to it, only recently some restaurants and top chefs have started using the *Food Pairing Hypothesis*. Thus, as shown in Figure 5(b), the GA was also able to find tastier combinations than those proposed by humans.

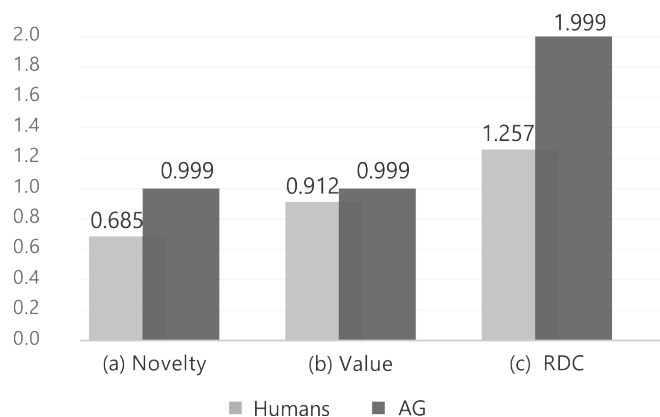


Figure 5: Average of novelty, value and creativity (RDC) of human-made recipes and recipes generated by genetic algorithm (GA) guided by the RDC metric.

Since it is possible to find new and valuable combinations, consequently, it is also possible to find more creative ones, which is shown in Figure 5(c). The use of the RDC metric as a fitness function allows the GA to prioritize combinations that have a balance among novelty and value.

Figure 2 shows the graph representing one of the most creative combination generated by *Creative Flavor Pairing*. It inspired the chef Otávio Mello from a restaurant in Brazil to create the recipe *Grilled Pork Loin with Guava Chutney and Beetroot and Potato Crispy*⁵. Otávio currently uses *Creative Flavor Pairing* to create his restaurant's menu. He indicated that the system helps him to explore surprising and tasty dishes.

Conclusions

The results show that the *Creative Flavor Pairing* is able to generate culinary ingredient combinations as creative as those generated by humans. It was also possible to verify that through the *RDC* metric, which guides the generation of artifacts optimizing novelty and value, the proposed system contributes to the culinary domain allowing the generation of combinations of ingredients little or never explored, minimizing the scenario described in (Ahn et al. 2011), where only a small portion of the area potential is exploited.

As future works, the proposed system can be expanded to use humans to evaluate if the combinations generated are really surprising and tasty. Another possibility would be to generate the full recipe, identifying in the quantities used of each ingredient, and generate the preparation directions. Regarding the structure of the recipe, another possibility would be to identify the distribution of ingredient categories in each recipe, and generate templates to be followed in the elaboration of the ingredient combinations.

⁵The *Grilled Pork Loin with Guava Chutney and Beetroot and Potato Crispy* recipe can be accessed at: <http://allrecipes.com/personal-recipe/64632063/>

References

- [Ahn et al. 2011] Ahn, Y.-Y.; Ahnert, S. E.; Bagrow, J. P.; and Barabási, A.-L. 2011. Flavor network and the principles of food pairing. *Scientific Reports* 1.
- [Amabile 1982] Amabile, T. M. 1982. The Social Psychology of Creativity: A Consensual Assessment Technique. *Journal of Personality and Social Psychology* 43(5):997–1013.
- [Baldi and Itti 2010] Baldi, P., and Itti, L. 2010. Of bits and wows: A bayesian theory of surprise with applications to attention. *Neural Networks* 23(5):649–666.
- [Besold, Schorlemmer, and Smaill 2015] Besold, R. T.; Schorlemmer, M.; and Smaill, A., eds. 2015. *Computational Creativity Research: Towards Creative Machines*. Paris: Atlantis Press.
- [Blumenthal 2008] Blumenthal, H. 2008. *The big fat duck cookbook*. Bloomsbury.
- [Boden 2004] Boden, M. A. 2004. *The Creative Mind: Myths and Mechanisms, Second Edition*. London: Routledge, 2 edition.
- [Boden 2009] Boden, M. a. 2009. Computer models of creativity. *AI MAGAZINE* 13(2):72–76.
- [Boden 2015] Boden, M. A. 2015. Creativity and ALife. *Artificial Life* 21(3):354–365.
- [Burdock 2016] Burdock, G. A. 2016. *Fenaroli’s handbook of flavor ingredients*. CRC press.
- [Cheng and Liu 2008] Cheng, C.-I., and Liu, D.-M. 2008. An intelligent clothes search system based on fashion styles. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 3, 1592–1597.
- [Colton et al. 2015] Colton, S.; Pease, A.; Corneli, J.; Cook, M.; Hepworth, R.; and Ventura, D. 2015. Stakeholder groups in computational creativity research and practice. In Besold, R. T.; Schorlemmer, M.; and Smaill, A., eds., *Computational Creativity Research: Towards Creative Machines*. Paris: Atlantis Press. 3–36.
- [Colton 2012] Colton, S. 2012. The painting fool: Stories from building an automated painter. In McCormack, J., and D’Inverno, M., eds., *Computers and Creativity*. New York: Springer.
- [Cook and Colton 2015] Cook, M., and Colton, S. 2015. Generating code for expressing simple preferences : Moving on from hardcoding and randomness. In Hannu Toivonen, Simon Colton, M. C. D. V., ed., *Proceedings of the Sixth International Conference on Computational Creativity*, 8–16. Provo: Brigham Young University.
- [França et al. 2016] França, C.; Góes, L. F. W.; Amorim, A.; Rocha, R.; and da Silva, A. R. 2016. Regent-dependent creativity: A domain independent metric for the assessment of creative artifacts. In François Pachet, Amílcar Cardoso, V. C. F. G., ed., *Proceedings of the Seventh International Conference on Computational Creativity*, 52–59. Paris: Sony Computer Science Laboratory.
- [Gabora 2010] Gabora, L. 2010. Revenge of the “Neurds”: Characterizing Creative Thought in Terms of the Structure and Dynamics of Memory. *Creativity Research Journal* 22(1):1–13.
- [Goel and Rugaber 2015] Goel, A. K., and Rugaber, S. 2015. Interactive meta-reasoning: Towards a cad-like environment for designing game-playing agents. In Besold, R. T.; Schorlemmer, M.; and Smaill, A., eds., *Computational Creativity Research: Towards Creative Machines*. Paris: Atlantis Press. 347–370.
- [Goes et al. 2016] Goes, L. F. W.; da Silva, A. R.; Rezende, J.; Amorim, A.; Franca, C.; Zaidan, T.; Olimpio, B.; Ranieri, L.; Morais, H.; Luana, S.; and Martins, C. A. P. S. 2016. Honingstone: Building creative combos with honing theory for a digital card game. *IEEE Transactions on Computational Intelligence and AI in Games* PP(99):1–7.
- [Grace and Maher 2016] Grace, K., and Maher, M. L. 2016. Surprise-triggered reformulation of design goals. In Dale Schuurmans, M. W., ed., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3726–3732. AAAI Press.
- [Grace et al. 2014] Grace, K.; Maher, M. L.; Fisher, D.; and Brady, K. 2014. Data-intensive evaluation of design creativity using novelty, value, and surprise. *International Journal of Design Creativity and Innovation* 3(3-4):125–147.
- [Jordanous, Allington, and Dueck 2015] Jordanous, A.; Allington, D.; and Dueck, B. 2015. Measuring cultural value using social network analysis: a case study on valuing electronic musicians. In Hannu Toivonen, Simon Colton, M. C. D. V., ed., *Proceedings of the Sixth International Conference on Computational Creativity*, 110–117. Provo: Brigham Young University.
- [Jordanous 2012] Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4(3):246–279.
- [Jordanous 2013] Jordanous, A. K. 2013. *Evaluating computational creativity: a standardised procedure for evaluating creative systems and its application*. Dotorado, Department of Informatics, University of Sussex, Brighton.
- [Joyner et al. 2015] Joyner, D. A.; Bedwell, D.; Graham, C.; Lemmon, W.; Martinez, O.; and Goel, A. K. 2015. Using human computation to acquire novel methods for addressing visual analogy problems on intelligence tests. In Hannu Toivonen, Simon Colton, M. C. D. V., ed., *Proceedings of the Sixth International Conference on Computational Creativity*, 23–30. Provo: Brigham Young University.
- [Kim and Cho 2000] Kim, H.-S., and Cho, S.-B. 2000. Application of interactive genetic algorithm to fashion design. *Engineering Applications of Artificial Intelligence* 13(6):635–644.
- [Kowaliw, Dorin, and McCormack 2012] Kowaliw, T.; Dorin, A.; and McCormack, J. 2012. Promoting creative design in interactive evolutionary computation. *IEEE Transactions on Evolutionary Computation* 16(4):523–536.
- [Krackhardt 1994] Krackhardt, D. 1994. Graph theoretical dimensions of informal organizations. In Carley, K. M., and

- Prietula, M. J., eds., *Computational Organization Theory*. Hillsdale: L. Erlbaum Associates Inc. 89–111.
- [Lamb, Brown, and Clarke 2015] Lamb, C.; Brown, D. G.; and Clarke, C. L. 2015. Human competence in creativity evaluation. In Hannu Toivonen, Simon Colton, M. C. D. V., ed., *Proceedings of the Sixth International Conference on Computational Creativity*, 102–109. Provo: Brigham Young University.
- [Macedo and Cardoso 2001] Macedo, L., and Cardoso, A. 2001. Modeling forms of surprise in an artificial agent. In Moore, J. D., and Stenning, K., eds., *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, 588–593. Edinburgh: Lawrence Erlbaum Associates.
- [Macedo, Reizenzein, and Cardoso 2004] Macedo, L.; Reizenzein, R.; and Cardoso, A. 2004. Modeling forms of surprise in artificial agents: empirical and theoretical study of surprise functions. In Kenneth Forbus, Dedre Gentner, T. R., ed., *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, 873–878. Mahwah: Lawrence Erlbaum.
- [Matta et al. 2016] Matta, C. F.; Sumar, I.; Cook, R.; and Ayers, P. W. 2016. Localization delocalization matrices and electron density weighted adjacency connectivity matrices: A bridge between the quantum theory of atoms in molecules and chemical graph theory. In Chauvin, R.; Lepetit, C.; Silvi, B.; and Alikhani, E., eds., *Applications of Topological Methods in Molecular Chemistry*. Switzerland: Springer International Publishing Switzerland. 53–88.
- [Morris et al. 2012] Morris, R. G.; Burton, S. H.; Bodily, P. M.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *International Conference on Computational Creativity*, 119–125.
- [Ortony and Partridge 1987] Ortony, A., and Partridge, D. 1987. Surprisingness and expectation failure: what’s the difference? In Kambhampati, S., ed., *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 106–108. São Paulo: AAAI Press.
- [Page and Dornenburg 2008] Page, K., and Dornenburg, A. 2008. *The flavor bible : the essential guide to culinary creativity, based on the wisdom of America’s most imaginative chefs*. New York: Little, Brown and Co.
- [Partridge 1985] Partridge, D. 1985. Input-expectation discrepancy reduction: A ubiquitous mechanism. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI’85*, 267–273. San Francisco: Morgan Kaufmann Publishers Inc.
- [Pinel, Varshney, and Bhattacharjya 2015] Pinel, F.; Varshney, L. R.; and Bhattacharjya, D. 2015. A Culinary Computational Creativity System. In Besold, T. R.; Schorlemmer, M.; and Smaill, A., eds., *Computational Creativity Research: Towards Creative Machines*, volume 7. Atlantis Press. chapter 16, 327–346.
- [Sawyer 2012] Sawyer, R. 2012. *Explaining creativity : the science of human innovation*. New York: Oxford University Press.
- [Sivanandam 2008] Sivanandam, S. N. 2008. *Introduction to genetic algorithms*. New York: Springer.
- [Stiensmeier-Pelster, Martini, and Reizenzein 1995] Stiensmeier-Pelster, J.; Martini, A.; and Reizenzein, R. 1995. The role of surprise in the attribution process. *Cognition & Emotion* 9(1):5–31.
- [Tomasic, Znidarsic, and Papa 2014] Tomasic, P.; Znidarsic, M.; and Papa, G. 2014. Implementation of a slogan generator. In Simon Colton, Dan Ventura, N. L. M. C., ed., *Proceedings of the Fifth International Conference on Computational Creativity*, 340–343. Ljubljana: Institut Jozef Stefan.
- [van der Velde et al. 2015] van der Velde, F.; Wolf, R. A.; Schmettow, M.; and Nazareth, D. S. 2015. A semantic map for evaluating creativity. In Hannu Toivonen, Simon Colton, M. C. D. V., ed., *Proceedings of the Sixth International Conference on Computational Creativity*, 94–101. Provo: Brigham Young University.
- [Varshney et al. 2013] Varshney, L. R.; Pinel, F.; Varshney, K. R.; Schorgendorfer, A.; and Chee, Y.-M. 2013. Cognition as a part of computational creativity. In *2013 IEEE 12th International Conference on Cognitive Informatics and Cognitive Computing*, 36–43. Institute of Electrical & Electronics Engineers (IEEE).
- [Varshney, Wang, and Varshney 2016] Varshney, L. R.; Wang, J.; and Varshney, K. R. 2016. Associative algorithms for computational creativity. *The Journal of Creative Behavior* 50(3):211–223.
- [Veeramachaneni et al. 2010] Veeramachaneni, K.; Vladislavleva, K.; Burland, M.; Parcon, J.; and O’ Reilly, U.-M. 2010. Evolutionary optimization of flavors. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO ’10*, 1291–1298. New York, NY, USA: ACM.
- [Wiggins 2006] Wiggins, G. A. 2006. Searching for computational creativity. *New Generation Computing* 24(3):209–222.
- [Witten, Frank, and Hall 2011] Witten, I. H.; Frank, E.; and Hall, M. a. 2011. *Data Mining Practical Machine Learning Tools and Techniques*. Burlington, MA, USA: Elsevier, 3 edition.

Creative Robot Dance with Variational Encoder

A. Augello, E. Cipolla, I. Infantino, A. Manfrè, G. Pilato, F. Vella

Institute for High Performance Computing and Networking

National Research Council of Italy (ICAR-CNR)

Via Ugo La Malfa 153

Palermo, 90146 Italy

Abstract

What we appreciate in dance is the ability of people to spontaneously improvise new movements and choreographies, sur-rendering to the music rhythm, being inspired by the current perceptions and sensations and by previous experiences, deeply stored in their memory. Like other human abilities, this, of course, is challenging to reproduce in an artificial entity such as a robot. Recent generations of anthropomorphic robots, the so-called humanoids, however, exhibit more and more sophisticated skills and raised the interest in robotic communities to design and experiment systems devoted to automatic dance generation. In this work, we highlight the importance to model a computational creativity behavior in dancing robots to avoid a mere execution of preprogrammed dances. In particular, we exploit a deep learning approach that allows a robot to generate in real time new dancing movements according to the listened music.

Introduction

The execution of artistic acts is certainly one of the most fascinating and impactful human activity that robotics aims to replicate. The abilities of new generation robots can be thoroughly tested in artistic domains, such as dance, music, painting and drama.

The appearance and the skills that characterize anthropomorphic robots make the dance domain very interesting and challenging since human movements can either be replicated, albeit imperfectly, or adapted to the embodiment of the robot. Dance is a harmonic composition of movements driven by the music stimuli and by the interactions with other subjects. Dancing movements follow the rhythm of the music and are synchronous with the song progression. Therefore both the timing and rhythm of the execution of the movements must be taken into account while trying to imitate human behavior.

The implementation of dancing capabilities in robots is not purely pursued for entertainment purposes. It provides new clues to deepen and improve various research themes, because it requires a robust learning phase that involves both a real-time analysis of music, the choice of harmonious and suitable movements, and moreover social behavior and interaction capabilities (Aucouturier et al. 2008; Augello et al. 2016; Shinozaki, Iwatani, and Nakatsu 2007).

The challenge lies in going beyond a preprogrammed dance executed by robots. A creative process should model

the mental processes involved in human creativity to generate movements and taking into account different music genres. The robots' perceptions should influence the choice processes and output of a learning process should lead to conceive a personal artistic style, that could be reconsidered or refined after the audience evaluation.

While some interesting and creative approach has been proposed for the generation of movements and choreographies (Jacob and Magerko 2015; Carlson et al. 2016; Crnkovic-Friis and Crnkovic-Friis 2016; Lapointe and Époque 2005), few are the works aimed at injecting a computational creativity behavior in dancing humanoids (Vircikova and Sincak 2010; Augello et al. 2016; Manfrè et al. 2016b).

In this work we explore the possibility for a robot to improvise a choreography, building on actions that are either stored in a memory or derived from a continuous elaboration of previous experiences. In our proposal, we took inspiration from human dance to create a dataset of movements that the robot can employ in his dance. The dataset, including also information about music features related to the sequences of movements, is used to train a variational encoder. This network allows obtaining a variation of the learned movements according to the listened music. The resulting movements are new but are coherent with the learned ones and are well synchronized with the listened music.

State of the art

Different works in literature propose approaches for dancing motions generation. One of these has been proposed and experimented by Luka and Louise Crnkovic-Friis (Crnkovic-Friis and Crnkovic-Friis 2016). It is a deep learning generative model, exploiting a Long Short-Term Memory type of recurrent neural network, that is used to produce novel choreographies. The network is trained on raw motion capture data, consisting of contemporary dance motions performed by a choreographer. The training dataset does not contain any information about musical features.

Cochoreo (Carlson et al. 2016) is a module of a sketching tool, named danceForms, used to create and animate keyframes. The module combines the functionality of a creativity support tool with an autonomously creative system, relying on a genetic algorithm, which generates novel keyframes for body positions. The new keyframes are eval-

uated according to a parameterized fitness function that allows the choreographer to set generation options based on their personal preferences.

Another example is the work proposed in (Aucouturier, Ogai, and Ikegami 2007), exploiting chaotic itinerancy (CI) dynamics generated by a network of artificial spiking neurons. The motions are chosen in real-time by converting the output of a neural network that processes the musical beats; then are executed by a vehicle-like robot.

For what concerns performances executed by humanoids, generally the interest is in the coordination of dancing gestures and postures according to the detected beats (Ellenberg et al. 2008; Grunberg et al. 2009; Seo et al. 2013; Shinozaki, Iwatani, and Nakatsu 2007)

Some creative approaches are discussed in (Augello et al. 2016; Infantino et al. 2016; Manfrè et al. 2016a; Vircikova and Sincak 2010; Eaton 2013; Xia et al. 2012). Zhou et al. have analyzed some of these works in the taxonomy of robotic dance systems discussed in (Peng et al. 2015). In addition to the already mentioned chaotic dynamic (Aucouturier, Ogai, and Ikegami 2007), they describe other approaches that have been proposed for the generation of dance choreographies. Meng et al. (Meng, Tholley, and Chung 2014), propose the use of the interactive reinforcement learning (IRL), to make robots learn to dance according to human preferences. Among the evolutionary computing approaches Zhou et al. (Peng et al. 2015), cite the algorithms proposed in (Eaton 2013; Vircikova and Sincak 2010). In detail, the authors of (Vircikova and Sincak 2010) have initialized a population of individuals encoded from dance characteristics. Then, the value of fitness of the algorithm is obtained with the interaction of the systems users, obtaining dances reflecting personal preferences. Another approach exploits the Markov Chain Model. As an example, in (Xia et al. 2012), each motion is considered as a Markov chain state, and the next motion is determined by the previous motion and the current music emotion. In our previous works (Infantino et al. 2016; Augello et al. 2016; Manfrè et al. 2016b) we proposed the use of both evolutionary computing and Markov models. We described the system underlying a humanoid dancing performance called ROBODANZA, also discussing the impact on different types of audience. In the performance, a humanoid robot interacts and dances with professional dancers, autonomously following the rhythms suggested by the dancers clapping the hands and tap on a table. The movements of the robot are generated according to a Hidden Markov model. Different emission matrices determine different execution styles of dance. The Transition Matrix (TM) of the HMM takes into account how a movement follows the previous one in a sequence of dance. It derives from observing the composition and occurrences of the human movement. The creative computational process exploits an interactive genetic algorithm that make the EMs to evolve according to the final user evaluations. Therefore each performance is always different.

Variational Autoencoder

Autoencoders have been enjoying significant interest as they can perform a lossy data compression starting from a spe-

cific dataset. Once trained, they represent in the hidden layer all the data have been previously exposed to; the representation is “lossy” since the reconstructed x is not perfectly identical to the original one, this difference being determined by the chosen distance or “loss” function.

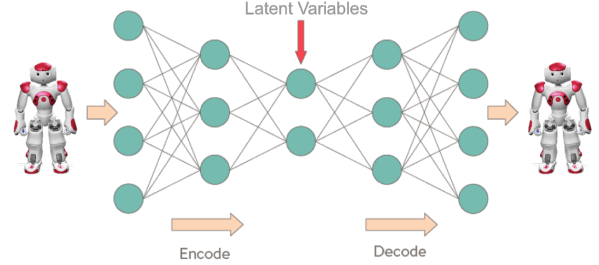


Figure 1: Architecture of Variational Autoencoder network. The network is able to faithfully reconstruct the input patterns.

As it can be seen in (Vitányi and Li 1997), compression and prediction are closely related fields, and compressors can also be used to generate new data.

Variational autoencoders, first introduced in (Kingma and Welling 2013), have raised much interest for their capability to produce a variation of the learned input data.

Their most interesting feature is the capability to autonomously draw the boundaries of a latent space in which the input data can be represented. Given input data x and calling $p(x)$ the probability distribution of the data, we want to learn the latent variable z with its probability density $p(z)$ so that data can be generated when the values of z are varied:

$$p(x) = \int p(x|z)p(z) \quad (1)$$

The training of the variational autoencoder is based on the variational inference to estimate the distribution $p(x|z)$. This method is often used in Bayesian methodology when you desire to infer a posterior that is difficult to compute. A simpler distribution $q_\lambda(z|x)$ is thus chosen as to minimize the Kullback-Leibler divergence between these two distributions. The variational parameter λ is used to refer to a family of distributions and, for a Gaussian family, would represent mean and variance. The divergence is calculated as:

$$D_{KL}(q_\lambda(z|x)||p(z|x)) = \mathbf{E}_q[\log \frac{q_\lambda(z|x)p(x)}{p(x,z)}] \quad (2)$$

It can be demonstrated that:

$$\log(p(x)) = L^v + D_{KL}(q_\lambda(z|x)||p(z|x)) \quad (3)$$

Since 2, to minimize the $\log(p(x))$ it is sufficient to minimize L^v . Its value can be calculated as

$$L^v = -D_{KL}(q(z|x)||p(z)) + \mathbf{E}_{q(z|x)} \log(p(x|z)) \quad (4)$$

Where the first term is $-D_{KL}(q(z|x)||p(z))$ representing the regularization part imposing the distribution of $p(z)$

as similar as possible to $q(z|x)$ while the second part $E_{q(z|x)} \log(p(x|z))$ takes into account a proper reconstruction of the values of x . After the training phase aimed at minimizing the value of $\log(p(x))$, that is equivalent to maximizing the likelihood, the values of zeta represent the best compression for the input values and variation in the z space corresponds to a variation in the reconstruction of input samples

Creative Robot Dance with variational encoder

Throughout this work the expression “variational encoder” is used to signify a change of the intended use of variational autoencoders; while the internal structure remains unchanged, latent variables are not used to allow a faithful reconstruction, but rather to introduce a different kind of information that proactively alters the reconstruction, enabling the robot to perform a different set of movements and also change its dancing style according to the past performances.

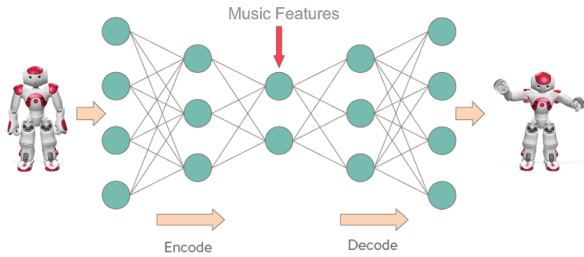


Figure 2: Architecture of Variational Encoder network. The network is able to reconstruct sequence of movement that can be varied giving the music features as an additional input.

The building blocks of a variational encoder are shown in Figure 4a. It is assumed that the information we deal with can be faithfully approximated with a Gaussian distribution, so that the encoder network can map the input samples into two parameters in a latent space, z_{mean} and $z_{log\sigma}$. They uniquely identify a given Gaussian distribution from which randomly sampled points z are then extracted. One of our contributions lies in the sampling function: altering the distinctive parameters of the Gaussian curve results in a different output mapping by the decoder network.

The parameters of the model are trained taking into account the reconstruction loss that forces the decoded samples to match the initial inputs. Furthermore, we minimize the Kullback-Leibler divergence between the learned latent distribution and the prior distribution, in order to avoid overfitting of the original dataset.

The decoding part, formed the hidden layer of the decoder and the decoder itself is shown in figure 4b and it is used for the prediction of the output movements. In our system, the creation of the robotic dance is based on the three processing phases: processing the sound, learning the movements and the generation of a sequence of movements.

The sound perceptions is implemented extracting some music features that represent the information in the listened music. The generation of the movements is based on the learning phase of the neural network. The execution is the combination of the conceived movements with the perceived music, synchronizing the motions with the rhythm. In the following sub sections details of the processes are given.

Learning phase

A basic understanding of the learning process in a human brain is needed when considering the same process in a neural network. It is believed that, during the learning process, the neural structure is altered by increasing or decreasing the strength of synaptic connections involved in a given activity. Artificial neural networks model this process by adjusting the weighted connections between neurons. Finding a satisfactory configuration may require several iterations which are collectively referred to as “training”.

In this work, a choreography is built around sequences of movements, that is, sequences of couples of poses. The dataset of joint values is partitioned into two subsets: the first one will be used to train the variational autoencoder, the second one for the prediction.

As detailed in previous sections, a correct choice of latent parameters is key to obtain a satisfactory reconstruction of the original input. Gaussian sampling is performed by taking into account both the loudness and the variance of a given music score; the mean value of the curve is the mean of loudnesses, and its standard deviation is the mean of music variances. A cycle of forward propagation of all the inputs and backward propagation of errors is called “epoch”. The number of training examples in one forward/backward pass is called “batch size”. In multi-layered networks backward propagation of errors for training is often used in conjunction with an optimization method.

In this work we used a variant of the stochastic gradient descent (SGD) optimization algorithm called Adadelta, first described in (Zeiler 2012). An extension to a previous algorithm called Adagrad (Duchi, Hazan, and Singer 2011), it adapts the learning rate to the frequency of parameters; in contrast to the original technique, only a fixed-size history of w squared gradients is considered, instead of the whole set of past gradients.

Let us call θ the parameters of the training set, $J(\theta)$ the objective function, and g_t the gradient of the objective function at time step t :

$$g_t = \nabla_{\theta} J(\theta) \quad (5)$$

To take history values into account, a running average over the gradient is introduced, depending only on the previous average and the current gradient:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) * g_t^2 \quad (6)$$

where $\gamma = 0.9$.

A SGD update can be described using the following equation:

$$\theta_{t+1} = \theta_t - \eta * g_{t,i} = \theta_t + \Delta\theta_t \quad (7)$$

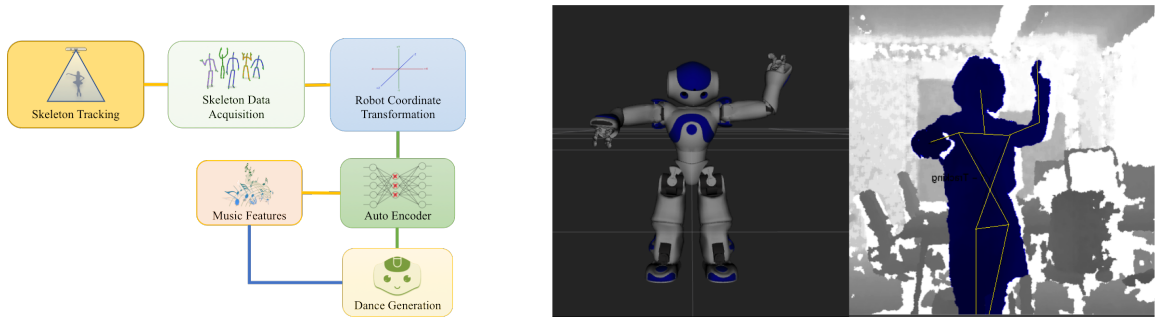
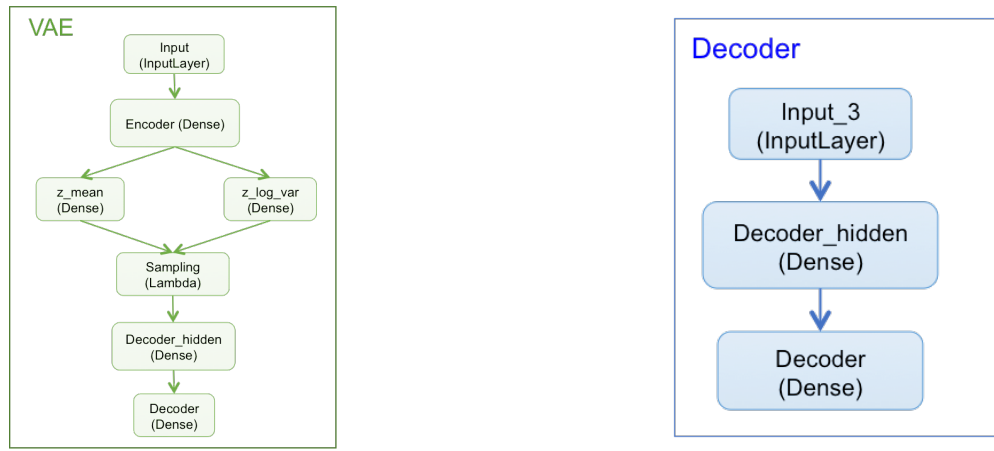


Figure 3: On the left side the schema of the learning phase used to obtain a set of dance movements by human demonstration is depicted. On the right there is the screenshot of the skeleton acquisition process during the learning phase and the corresponding posture of the simulated robot.



(a) End-to-end model of the variational autoencoder

(b) Detailed view of the decoder/predictor

Figure 4: Block diagrams for the variational autoencoder

where $\Delta\theta_t$ is the parameter update vector.

It can be demonstrated that the parameter update vector can be rewritten as follows:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}g_t = -\frac{\eta}{RMS[g]_t}g_t \quad (8)$$

If the decaying average over squared parameter updates is defined as:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma) * \Delta\theta^2 \quad (9)$$

the update rule in a way that is not dependent on the learning rate η :

$$\theta_{t+1} = \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t}g_t \quad (10)$$

Robotic Dance

In order to have a dataset of variegated movements, during multiple sessions we have recorded and stored the movements of four users having different experience. The dataset is composed of both slightly harmonic repetitive sequences

and movements with high variability and harmonicity. The different dance experience and the multiple session of the users provide a diversified dataset since a human usually do not use the same sequences of movements during an improvised dance with music. The heterogeneity of the dataset is due to the different level of dance experience of the users. In our opinion, all these movements may contribute to creating new unseen movements mixing the learned samples.

The generation of the movements starts with the extraction of some feature from the music input, in particular, the system extracts the loudness and the variance of the rhythmic sound that will be used as input of the latent space of the variational autoencoder to generate the movements according to the perceived music.

Since the latent space of the network is Gaussian, the values of loudness and variance have to be transformed through the inverse cumulative distribution function of the Gaussian to produce coherent values with the latent space. The transformed value are the real input of the network's latent space and these value lead the generation of the movement related to the perceived music.

Our idea is to allow the robot to execute one motion ac-

ording the beat. Given the position of the beat and the interval between two consecutive beats, the system can execute one movement for each detected interval. The network, using as an input the intensity and variance of the music interval outputs a configuration of joints that represents one single movement. The numbers of movements that the network predicts is function of the number of processed music features.

The final part of the system focuses on the execution of the movements by the robot in synchrony with the music. The execution of the dance is made up combining the movements predicted by the network and the features of the audio signal to keeping time, in fact, we use the information of the beat position and beat interval to regulate the duration of each movement.

The proposed system is flexible and can adapt to different music genres, in fact, whatever type of rhythmic music is provided as an input, the system can generate sequences of movements. Moreover, it is possible to continue to train the network with others music genres, adding new movements captured in different learning sessions with human dancers.

Experimental Results and Discussion

In the following subsections we describe the experimental setup, the results and a brief discussion about evaluation issues of the system.

Dance Movement Acquisition

To collect the dataset of the movements, we employed the Microsoft RGB-D¹ Kinect camera to track the improvised movements of the human dancers. We used the Kinect camera since it is non-invasive and cheap compared to other motion capture systems, avoiding to use high precision capture devices. Using a Kinect camera the dancer does not need to wear any device and he can act in a natural manner; moreover, considering that the robot cannot reproduce all the possible human posture due to its structural limitation more precision does not resolve such problem.

The Kinect includes a RGB camera and four microphones to capture sounds, an infrared (IR) emitter and an IR depth sensor to capture the depth map of each image. The collected information enables the extraction of the spatial position of the dancer's joints in a non-invasive way. The advantage in employing the kinect is that any person in a room can be recorded as a teacher for the dance without the any preparation and without setting the connections that are required for the body sensors.

The use of Kinect camera allows capturing graceful and pleasant movements displayed in front of the acquisition system and the good sampling frequency, around thirty frame per second, allows to maintain a high correlation between a motion and the next one.

To interface the acquisition device with the NAO robot, that is used to reproduce natural motion, we used ROS (Robotics Operating System)². It is an open-source operating system for robots that provides a layered structure to

communicate in a peer-to-peer topology between server and hosts. ROS allows the user to connect different hosts at runtime, managing messages among processes, sensors and actuators.

Through the Kinect sensor it is possible to extract the skeleton data of a human dancer, obtaining in real-time the list of the 15 joint position of the human body along the three axis (x,y,z). The skeleton information cannot be directly used for the robot positions since the coordinate system are different and the robot has several limitations if compared to human movements. Hence, the skeleton data extracted should be transformed to change the coordinate system to perform the movements in the robot. To track the position of the arms and the head during the motion of the dancer we have used the ROS package *skeleton markers*³. In (Rodriguez et al. 2014) is described a system, based on the same technologies, to set up a robust teleoperation system. In the current implementation a stronger focus has been given to the movement of the upper body part not to compromise the stability of the robot during the dance.

The dataset of the movements has been created by observing the dance of four different people; *User 1* and *User 2* have a limited competence in the dance field. *User 3* owns an expertise in dance, she is not a professional but often performs dance and has a good sense of rhythm. *User 4* is a professional in couple dancing, highly skilled to follow the rhythm. They were asked to execute spontaneous motions while listening to different songs, while the RGB-D camera of the Microsoft Kinect v1 acquired the sequence of their movements and saved photo shots.

The system tracks and samples the movements of the human dancers as soon as music beats are detected. After the conversion in the robot coordinate system, the transformed joint positions are stored to be subsequently used during the training phase. Since the robot has less degrees of freedom than human beings, some complex movements appear to be truncated if compared with the original ones. For example, the robot is not able to perform sinusoidal movements with the arm or push forward its shoulder.

Sound Processing

The Essentia Library (Bogdanov et al. 2013) has been used to identify music features that will be used to generate movements, enabling the robot to dance in synchrony with the music.

The following features are extracted: Beats location, Beat Per Minute (BPM), beat interval, variance and loudness.

Beat locations give the timestamps where the rhythm falls, whereas beat intervals indicate the interval of time between two consecutive beats; both are used to compute the intensity and the variation within two subsequent musical beats and to synchronize the movements with the music.


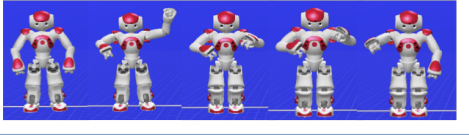
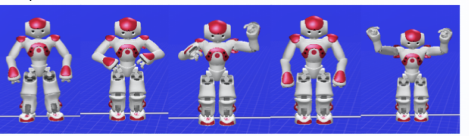
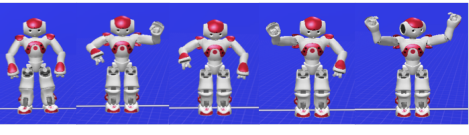
From the beat positions we calculate the loudness and the variance within the frame between two consecutive beats that are used in the successive steps to generate movements by mean the network.

¹Red Green Blue plus Depth

²<http://wiki.ros.org/it>

³wiki.ros.org/skeletonmarkers

Table 1: Evaluation of Robot movements versus the learning epochs

<i>Poses</i>	<i>Comments</i>
<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p>5 Epoch</p>  </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p>10 Epoch</p>  </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p>20 Epoch</p>  </div> <div style="border: 1px solid black; padding: 5px;"> <p>25 Epoch</p>  </div>	<p>The robot basically repeats the same simple movements.</p> <p>There is some more variability. The left arm is risen independently.</p> <p>Much more variability. Movements become more complex and do not seem to follow a pre-recorded pattern.</p> <p>Maximum variability. Uses both new and old poses. Even the head starts moving.</p>

Dancing Experiments

Robotic dance information has been acquired using a set of custom Python scripts. The autoencoder has been implemented using the Keras⁴ (Chollet 2015) open source framework for rapid prototyping of deep networks. To train the model four datasets are needed:

- Two sets P_1, P_2 containing k joint poses.
- Music variances V
- Music loudnesses L

The mean of values in V, V_m , and the mean of values in L, L_m , are calculated; they will be used to create the Gaussian distribution used to sample latent features.

Successive joint poses in P_1 and P_2 are coupled to form two sets M_1 and M_2 containing $k/2$ movements. The set M_2 is M_1 forward shifted of 1 time unit. Movements in M_1 and M_2 will be supplied as input and as expected output, respectively, to train the encoder.

When training process has converged new movements can be generated providing values of variance and loudness values to the latent units of the network. If the values are ex-

tracted from the musical piece, the robot can improvise a dance following the features of the listened song.

Table 2 shows the variance of generated joint values as the requested number of epochs increases up to the 25th, which is the last we have considered, as seen in , as further runs do not produce an appreciable decrease in the error function.

The variance of joint configurations measures how far robot movements deviate from the mean; higher values of variance may thus be used to signify an increased creativity in motion sequence generation. This insight is substantiated if we consider some example movements generated at different epochs, as shown in the table 1.

While at epoch 1 movement sequences are quite repetitive, they become more and more harmonic as the neural network continues the training. At epoch 25 a remarkable distinctiveness can be detected.

We have also computed mean of the variance of the joint value during different epochs. An interesting consideration to be taken into account is that the trend of variance stalls between 10 and 20 epochs, and then reaches a maximum at epoch 25. In Figure 5 are shown the plot of the angle values (in radiant) of the right and left joint in the shoulder and elbow. The value are referred to epoch 25 and show a rich set of movements learned at the final steps.

⁴<http://keras.io/>

Table 2: Variance of the joints

<i>Names of Joints</i>	<i>Epoch 1</i>	<i>Epoch 5</i>	<i>Epoch 10</i>	<i>Epoch 15</i>	<i>Epoch 20</i>	<i>Epoch 25</i>
LElbowRoll	0.0584	0.8002	0.1806	0.1651	0.1522	0.3280
RElbowRoll	0.0390	1.0252	0.4269	0.1781	0.1924	0.3049
LElbowYaw	0.2596	0.2529	0.2095	0.1018	0.6654	0.3256
RElbowYaw	0.2219	0.3678	0.2021	0.3635	0.1965	0.1898
LShoulderRoll	0.1045	0.1724	0.2383	0.0376	0.3827	0.4827
RShoulderRoll	0.0253	0.1207	0.2563	0.3597	0.5258	0.2854
LShoulderPitch	0.3577	0.1392	0.1505	0.1388	0.3065	0.7498
RShoulderPitch	0.1993	0.1547	0.2927	1.2272	1.1995	1.4630
HeadYaw	0.0416	0.0980	0.0539	0.0306	0.0176	0.0152
HeadPitch	0.0001	0.0002	0.0003	0.0000	0.0001	0.0002

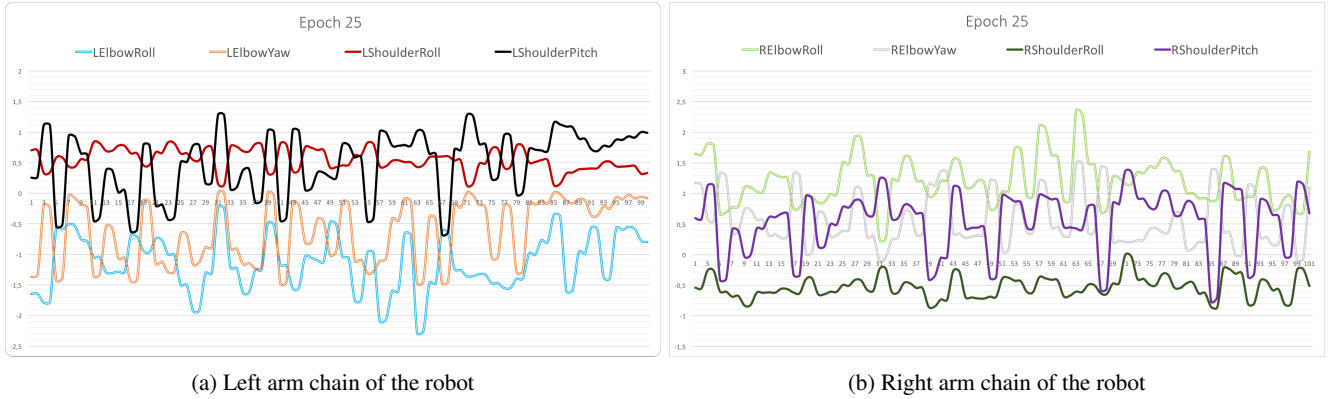


Figure 5: Progress of the joints values during a dance session

Table 3: Mean of the variance of the joint value during different epochs

	<i>Variance Mean</i>
<i>Epoch 1</i>	0.1307
<i>Epoch 5</i>	0.3131
<i>Epoch 10</i>	0.2011
<i>Epoch 15</i>	0.2603
<i>Epoch 20</i>	0.3639
<i>Epoch 25</i>	0.4145

In our opinion the evaluation of the output of artificial systems is a key issue for well founded computational creativity. The implementation of the dance with neural encoders tends to focus on a restricted set of movements and to repeat the same patterns. From this point of view the variance of the movements is a key parameter indicating that a large set of movements has been learned and the dance resembles human movements. Furthermore an external evaluation from people staring at the dance performance can be used to select the most adapted movements for a robotic dance. In previous work (Manfrè et al. 2017), we used a clustering approach to define groups collecting similar dance actions. Within the cluster, each movement has an evaluation score initially set to 100. The centroid of the cluster is the first representative of the group used to determine the dance creation

as previously described. If a user evaluates the performance as negative, then the scores of the movements belonging to the sequence are lowered. After hundreds of performances, when a movement has a score below a given threshold (e.g. less than 50), the system searches for the substitute with the highest score in the same cluster. The positive evaluation causes an increment of the related scores of involved movements. The judgment of an expert evaluator provides another evaluation mechanism that has a strong influence on dance execution. In fact, the expert could indicate inadequate a single movement or a short sequence and directly causes an inhibition (i.e. the score is equal to 0).

In (Augello et al. 2016) we report the evaluation results of various live performances with heterogeneous audiences. Even we have no rigorous experimental evidence the results seem to demonstrate that variability of movement is an important factor for a positive evaluation.

Conclusions

In this work, we proposed a deep learning approach to induce a computational creativity behavior in a dancing robot. In particular we used a variational encoder that allows mapping input patterns in a latent space.

The encoder has been trained with a set of movements captured from differently skilled dancers. The generation is obtained by injecting the representation of the listened music in the latent space of the encoder network. As a result,

the robot is able to improvise dancing movements according to the listened music even if it has not been previously presented in the learning phase.

References

- Aucouturier, J.-J.; Ikeuchi, K.; Hirukawa, H.; Nakaoka, S.; Shiratori, T.; Kudoh, S.; Kanehiro, F.; Ogata, T.; Kozima, H.; Okuno, H. G.; et al. 2008. Cheek to chip: Dancing robots and ai's future. *IEEE Intelligent Systems* 23(2).
- Aucouturier, J.-J.; Ogai, Y.; and Ikegami, T. 2007. Making a robot dance to music using chaotic itinerancy in a network of fitzhugh-nagumo neurons. In *International Conference on Neural Information Processing*, 647–656. Springer.
- Augello, A.; Infantino, I.; Manfrè, A.; Pilato, G.; Vella, F.; and Chella, A. 2016. Creation and cognition for humanoid live dancing. *Robotics and Autonomous Systems* 86:128–137.
- Bogdanov, D.; Wack, N.; Gómez, E.; Gulati, S.; Herrera, P.; Mayor, O.; Roma, G.; Salamon, J.; Zapata, J. R.; and Serra, X. 2013. Essentia: An audio analysis library for music information retrieval. In *ISMIR*, 493–498. Citeseer.
- Carlson, K.; Pasquier, P.; Tsang, H. H.; Phillips, J.; Schiphorst, T.; and Calvert, T. 2016. Cochoreo: A generative feature in idanceforms for creating novel keyframe animation for choreography. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Chollet, F. 2015. keras. <https://github.com/fchollet/keras>.
- Crnkovic-Friis, L., and Crnkovic-Friis, L. 2016. Generative choreography using deep learning. *arXiv preprint arXiv:1605.06921*.
- Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12:2121–2159.
- Eaton, M. 2013. An approach to the synthesis of humanoid robot dance using non-interactive evolutionary techniques. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, 3305–3309. IEEE.
- Ellenberg, R.; Grunberg, D.; Kim, Y.; and Oh, P. 2008. Exploring creativity through humanoids and dance. In *Proceedings of The 5th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI*.
- Grunberg, D.; Ellenberg, R.; Kim, Y.; and Oh, P. 2009. Creating an autonomous dancing robot. In *Proceedings of the 2009 International Conference on Hybrid Information Technology*, 221–227. ACM.
- Infantino, I.; Augello, A.; Manfrè, A.; Pilato, G.; and Vella, F. 2016. Robodanza: Live performances of a creative dancing humanoid. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Jacob, M., and Magerko, B. 2015. Viewpoints ai. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 361–362. ACM.
- Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Lapointe, F.-J., and Époque, M. 2005. The dancing genome project: generation of a human-computer choreography using a genetic algorithm. In *Proceedings of the 13th annual ACM international conference on Multimedia*, 555–558. ACM.
- Manfrè, A.; Augello, A.; Pilato, G.; Vella, F.; and Infantino, I. 2016a. Exploiting interactive genetic algorithms for creative humanoid dancing. *Biologically Inspired Cognitive Architectures* 17:12–21.
- Manfrè, A.; Infantino, I.; Vella, F.; and Gaglio, S. 2016b. An automatic system for humanoid dance creation. *Biologically Inspired Cognitive Architectures* 15:1–9.
- Manfrè, A.; Infantino, I.; Augello, A.; Pilato, G.; and Vella, F. 2017. Learning by demonstration for a dancing robot within a computational creativity framework. In *Proceedings of the First IEEE International Conference on Robotic Computing, ICRC 2017, Taiwan*.
- Meng, Q.; Tholley, I.; and Chung, P. W. 2014. Robots learn to dance through interaction with humans. *Neural Computing and Applications* 24(1):117–124.
- Peng, H.; Zhou, C.; Hu, H.; Chao, F.; and Li, J. 2015. Robotic dance in social robotics taxonomy. *IEEE Transactions on Human-Machine Systems* 45(3):281–293.
- Rodriguez, I.; Astigarraga, A.; Jauregi, E.; Ruiz, T.; and Lazkano, E. 2014. Humanizing nao robot teleoperation using ros. In *Humanoids*, 179–186.
- Seo, J.-H.; Yang, J.-Y.; Kim, J.; and Kwon, D.-S. 2013. Autonomous humanoid robot dance generation system based on real-time music input. In *RO-MAN, 2013 IEEE*, 204–209. IEEE.
- Shinozaki, K.; Iwatani, A.; and Nakatsu, R. 2007. Concept and construction of a robot dance system. In *International Workshop and Conference on Photonics and Nanotechnology 2007*, 67942J–67942J. International Society for Optics and Photonics.
- Vircikova, M., and Sincak, P. 2010. Dance choreography design of humanoid robots using interactive evolutionary computation. In *3rd Workshop for Young Researchers: Human Friendly Robotics for Young Researchers*.
- Vitányi, P., and Li, M. 1997. On prediction by data compression. In *European Conference on Machine Learning*, 14–30. Springer.
- Xia, G.; Tay, J.; Dannenberg, R.; and Veloso, M. 2012. Autonomous robot dancing driven by beats and emotions of music. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 205–212. International Foundation for Autonomous Agents and Multiagent Systems.
- Zeiler, M. D. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701.

Text Transformation Via Constraints and Word Embedding

Benjamin Bay, Paul Bodily, and Dan Ventura

Computer Science Department
Brigham Young University
Provo, UT 84602 USA

benjamin.bay@gmail.com, paulmbodily@gmail.com, ventura@cs.byu.edu

Abstract

In order to provide resources for artistic communities and further the linguistic capabilities of computationally creative systems, we present a computational process for creative text transformation and evaluation. Its purpose is to help solve the fundamental problem posed by the field of natural language generation, which is to computationally generate human-readable language. Our process entails the use of 1) vector word embedding to approximate meaning and 2) constraints to guide word replacement. We introduce *intentions* as objects that drive the generation of creative artefacts; a target theme, emotion, meter, or rhyme scheme may be represented via intention. Our implementation of this process, *Lyrlist*, is oriented around poetry and song lyrics and successfully produces syntactically correct, human-voiced text. A preliminary evaluation suggests that our process successfully evokes human-recognizable sentiments and that even familiar texts are difficult to recognize after undergoing transformation.

Introduction

Language is an incredible tool used by humans. It expresses our most complex ideas and is woven into all of our creative tasks; the mediums of conversation, poetry, and song are built around it. As such, gaining power over language is integral to the problem of computational creativity (CC); two important goals are computational 1) understanding of language and 2) production of language (Bateman and Zock 2003). Natural language processing (NLP) and its subfield, natural language generation (NLG), address these problems.

Syntactically correct text is available in abundance and is easily produced by humans. However, maintaining such syntactic correctness and semantic cohesion in generative text is a major barrier within NLG. We bypass this challenge by transforming, rather than generating, texts. This allows us to modify individual words of an original text while maintaining its relative word relationships and original syntactical structure. In this paper we present a general framework for systems to 1) transform and 2) evaluate textual artefacts.

We present an implementation of this transformation framework, applied to the problem of generating poetry and song lyrics, as well as a protocol for creative intention.

Related Work

Our method of text transformation builds upon ideas from previous research in computational linguistics and CC.

Monteith et al. presented a successful process for creating melodic accompaniments based on input lyrics (2012). This approach to music generation assumes the availability of pre-existing lyrics. Our transformation process fills the essential role of lyric-generation for such systems.

Exploring the problem of automatic lyric generation, Oliveira, et al. built a system that generates lyrics with word stresses matching the rhythm of given melodies (2007). Later, Oliveira improved this system to generate text on a semantic domain using seed words (2015). For the sake of autonomy, we wanted to design a framework that did not necessarily rely on syllabic stress data to generate text.

Tobing et al. approached the problem of poetry generation via grammars and chart generation (2015). While their results are, for the most part, syntactically correct, they are not semantically cohesive, and their method is described as “a work in progress”. Because of the complexity of generating speech via grammars, our framework currently avoids them altogether.

Colton et al. build a poetry generator that used poem templates and simile tuples of the form $\langle object, aspect, description \rangle$ (2012). While their simile knowledge base is high-quality, we seek a more comprehensive approach with regard to comparisons; in other words, we seek the ability to quantify the relationship between any set of words in the English language—by leveraging vector-based word embeddings for managing simile, analogy, and metaphor.

Toivanen et al. replace 50% of words in lyrical templates derived from existing songs (2013). Their method is simple, effective, and presents only low risk of losing morphological or global semantic cohesion. However, it only achieves limited transformation and limited creativity because it excludes prioritized constraints, the notion of intention, and the ability to estimate word meaning.

Hirjee and Brown use a confusion matrix of probabilities that any pair of phonemes will rhyme based on rhymes found in a corpus of hip-hop lyrics (2009). Such a matrix could be used to score potential rhymes between words; this is an intelligent approach. However, these probabilities are empirically derived, and we seek a more principled rhyme-

Input	Output
$\sigma(\text{walking})$	running, stepping, hiking
$\tau(\text{actress, mom, aunt, queen})$	female, girl, woman
$\alpha(\text{house, castle, roof})$	parapet, battlements, spire

Table 1: Examples of word-vector operations made possible by word embedding. The spatial distribution of words in such vector spaces is such that nearby words are related, and basic vector arithmetic allows for computational approximation of word meaning. These operations accurately reveal basic relationships, such as “walking” being related to “hiking”, or “female” being the common thread between “actress”, “mom”, “aunt”, and “queen”; they also reveal interesting and imagerial relationships, such as the roof of a castle being a “parapet” or “battlements”.

scoring algorithm rooted in phonology.

Gervás et al. discuss the challenges of automatic poem generation, and offer these classes of solution: understanding phonetics, using phonetic knowledge to drive poem generation, managing vocabulary, dealing with emotions, and managing comparison, analogy, and metaphor. (2007). We agree that these are among the primary challenges of poem generation (and more broadly, text transformation) and have implemented solutions in each case.

Finally, Gervás groups approaches to poetry generation into these rough categories: 1) template-based, 2) generate-and-test, 3) evolutionary, and 4) case-based reasoning (2002). Though not exclusive to poetry applications, our approach best fits into the first two categories. Rather than generate text, our framework transforms and evaluates it.

Framework

Our conceptual framework for the creative transformation of text includes a language model, a system of prioritized constraints, a text transformation process, and a protocol for computational intention. The language model is used to find appropriate replacements for portions of the text; the constraints filter these potential replacements and impose the intentions on the transformation process.

Language Model (\mathcal{L})

The language model \mathcal{L} is effected as a word embedding, which entails mapping words (usually from a large textual corpus) to vectors of real numbers in a multi-dimensional space such that collocated words are semantically related. More formally, $\varepsilon : W \rightarrow \mathbb{R}^n$, with W the set of all words and n the dimensionality of the embedding space.

Once such a vector space is constructed, the estimation of word meanings and relationships becomes possible using geometric operations, and the primary operations used here for text transformation are similarity σ , neighbor χ , theme τ , and analogy α . These are described in detail below and examples are provided in Table 1.

The similarity operator $\sigma : W^2 \rightarrow [0 \dots 1]$ is defined as

$$\sigma(w_1, w_2) = \frac{|\varepsilon(w_1) \cdot \varepsilon(w_2)|}{\|\varepsilon(w_1)\| \|\varepsilon(w_2)\|} \quad (1)$$

It computes the absolute value of the *cosine similarity* between the vectors associated with two words w_1, w_2 , which, given the structure of the embedding space, is a good geometric surrogate for similarity of word meaning.

The neighbor operator $\chi : W \rightarrow 2^W$ computes a set of neighbors for a word w and is defined as

$$\chi(w) = \{y | v \in \mathbb{R}^n \wedge \sigma(v, \varepsilon(w)) < \theta \wedge y = \varepsilon^{-1}(v)\} \quad (2)$$

where θ is a threshold for controlling the number of neighbors and $\varepsilon^{-1} : \mathbb{R}^n \rightarrow W$ is a partial inverse function that extracts words from the embedding space. χ returns a set of words mapped to points in the space that are close to the point to which w is mapped. These neighbor words generally have similar usages as or associations with w .

The theme for a set of words Y is computed with the operator $\tau : 2^W \rightarrow W$, which is defined as

$$\tau(Y) = \varepsilon^{-1} \left(\zeta \left(\frac{1}{|Y|} \sum_{i=1}^{|Y|} \varepsilon(y_i) \right) \right) \quad (3)$$

where $y_i \in Y$, and $\zeta : \mathbb{R}^n \rightarrow \mathbb{R}^n$ maps vectors into the domain of ε^{-1} . That is,

$$\zeta(v) = \operatorname{argmax}_{u \in \varepsilon[W]} \sigma(u, v) \quad (4)$$

where the notation $f[A]$ means the image of the set A under f (so, we are looking for vectors in the image of ε so that the inverse mapping ε^{-1} will be defined). τ finds the word whose vector embedding is closest to the centroid of all the vector embeddings of the words in Y ; that is, it finds the “average” of the set Y , effectively summarizing the text represented by Y .

The analogy operator $\alpha : W^3 \rightarrow W$ is defined as

$$\alpha(a, b, c) = \varepsilon^{-1}(-\varepsilon(a) + \varepsilon(b) + \varepsilon(c)) \quad (5)$$

Because of the semantic structure of the embedding space, the output of this function is often a reasonable completion d for the analogical form $a : b :: c : d$.

Instead of training a language model (vector embedding) for every possible genre, dialect, and time period, we train one master model and filter its word suggestions with constraints.

Constraints ($\mathcal{C} = \langle \mathcal{C}_m, \mathcal{C}_r \rangle$)

Two types of constraints are used to filter potential solutions for the transformation process. Marking constraints, \mathcal{C}_m , determine which words in a text will be transformed, and replacement constraints, \mathcal{C}_r , determine which candidate replacements are valid. In practice, both types of constraints may be implemented as a conjunction of Boolean predicates, $\mathcal{C}_m = \{P_i^m\}$ and $\mathcal{C}_r = \{P_j^r\}$.

As an example, consider the case of marking constraints composed of single compound predicate $\mathcal{C}_m = \{P^m\}$, where

$$P^m(w) = \begin{cases} \text{TRUE} & \text{noun}(w) \vee \text{verb}(w) \vee \text{adj}(w) \\ \text{FALSE} & \text{otherwise} \end{cases}$$

that picks out those words that are either nouns, verbs or adjectives. Then, if the set of replacement constraints \mathcal{C}_r were the following (in priority order):

- $$\mathcal{C}_r = \left\{ \begin{array}{l} 1. \text{ spelling} \neq \text{ original spelling} \\ 2. \text{ lemma is not to equal to any other new} \\ \quad \text{lemma in the transformation} \\ 3. \text{ not on restricted list} \\ 4. \text{ found in an English dictionary} \\ 5. \text{ part of speech is identical to original's} \\ \quad \text{part of speech} \end{array} \right\}$$

then for each noun/verb/adjective, a replacement word must be found that meets all of the criteria listed above: it must be a different word, must not have already been used, must not be blacklisted, must be English, and must have the same part of speech.

Transformation (\mathcal{T})

Given a text T to be transformed, the set of marking constraints \mathcal{C}_m determines which words $M \subseteq T$ will be replaced. For each word $m \in M$ to be replaced, the language model \mathcal{L} is used to find a set of potential replacement words R , and each suggestion $r \in R$ is then filtered with a prioritized list of replacement constraints \mathcal{C}_r . Thus, the transformation $\mathcal{T}(T) = T'$ is computed as follows

$$M = \{t | t \in T \text{ and } \bigwedge_i P_i^m(t)\}$$

and for $m \in M$

$$R_m = \mathcal{L}(m)^1$$

and for $r \in R_m$

$$S_r = \{r | r \in R_m \text{ and } \bigwedge_i P_i^r(r)\}$$

In the case that $S_r = \emptyset$, replacement constraints can be weakened or dropped in reverse order of their priority until the set is no longer empty. In the case that $|S_r| > 1$, one member of S_r can be chosen probabilistically. The final result is the transformed text T' in which each marked word $m \in T$ has been replaced by a word m' that fits the language model \mathcal{L} and meets the constraints in \mathcal{C}_r .

Intention (\mathcal{I})

Just as an artist puts careful consideration into a creative task before it is carried out, creative systems may possess objectives which influence the eventual creation of artefacts. We call such objectives *intentions*. For our purposes, an intention has the following properties:

¹We abuse notation here to mean that one or more operators associated with \mathcal{L} are applied to m .

- Determinative - may be used to direct the creation of an artefact
- Evaluative - may be used to evaluate an artefact
- Selective - may be applied globally or locally within an artefact
- Conjunctive - may be combined with other intentions

We identify the following three classes of intention as applicable in the artefact generation context:

- Thematic intention (\mathcal{I}_t) - the semantic purpose of the artefact (e.g., subject, emotion).
- Cultural intention (\mathcal{I}_c) - the sociocultural context for the artefact (e.g., language, movement, genre).
- Structural intention (\mathcal{I}_s) - the target organization or arrangement of an artefact (e.g., technique, rhyme scheme, meter).

A tuple of sets corresponding to the above three classes may be used to represent overall intention for an artefact:

$$\mathcal{I} = \langle \mathcal{I}_t, \mathcal{I}_c, \mathcal{I}_s \rangle. \quad (6)$$

Intention can be imposed on the system through design decisions (e.g., which corpora to use in training the language model \mathcal{L}), additional replacement constraints (e.g., rhyming, syllable count), parameter selection (e.g., thematic seed words to be used with the α operator), etc.

Inspiration refers to the method or source from which intention is determined: a system with immutable intentions determined by the designer may produce interesting artefacts, but it will be considered less autonomous (and thus likely less creative) than a system whose intentions are mutable and that defines its intentions via some other form of inspiration.

Implementation

We built a poem and lyric transformer as a proof of concept for the transformation framework, and named it *Lyrlist*.

We used Word2Vec's continuous bag of words learning model (Mikolov et al. 2013) to construct the embedding for our language model \mathcal{L} . We used a dimensionality of $n = 500$, and a window size of 5. Figure 1 shows the composition of the corpus used for training the model, by type of text. We sought to maximize the quantity of artistic and spoken texts, and use a large quantity of text overall, deriving our corpus from a data set of pop song lyrics, a data set of poems, COCA (Davies 2008), the NOW Corpus (Davies 2016), and Wikipedia.

In *Lyrlist*, each word object carries the attributes listed in Table 2. We use Stanford CoreNLP (Manning et al. 2014) for part of speech tagging, named entity tagging, and lemmatization. Parts of speech are drawn from the Penn Treebank tag set (Marcus, Marcinkiewicz, and Santorini 1993). We use the CMU Pronouncing Dictionary (Kominek and Black 2004) to assign words phonemes and stresses. Phonemes are represented by ARPAbet, a phonetic transcription code designed specifically for English.

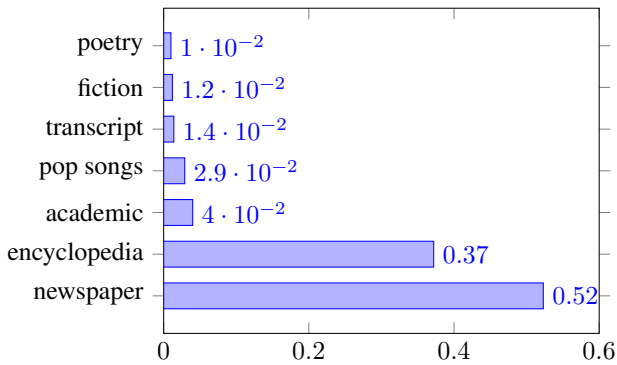


Figure 1: Proportions of text types used in the language model \mathcal{L} of Lyryst. We consider creative texts (i.e., poetry, song lyrics, novels) and transcripts of actual human conversations to be the best training corpora for artistic text transformation purposes. However, these types of text are difficult to accumulate on a large scale, so we also include texts more accessible in bulk, such as academic papers, magazines, newspapers, and encyclopedias.

Lyryst makes use of both thematic and structural intentions; we currently implement the cultural intention of “English” only, leaving a cultural database and a more thorough treatment of intention for future work. Its intentions are currently human-defined, and we leave the implementation of a system on inspiration also for future work. To support the structural intentions, we implemented both a syllable parser and a rhyme scorer, in order to gain control over meter and rhyme. Here we provide details on the specifics of our implementation.

Syllabification A word is a sequence of syllables. A syllable is made of an onset ω , nucleus ν , and coda κ . The nucleus is the central vowel phoneme. The onset is the con-

Spelling	“Saturdays”
Lemma	“Saturday”
Phonemes	S-AE-T-ER-D-AY-Z
Syllables	S-AE-T · ER · D-AY-Z
Stresses	1 · 0 · 0
Stress tail	AE-T · ER · D-AY-Z
Part of speech tag	NNS
Named entity tag	WEEKDAY

Table 2: Example of useful data about the word $w = \text{Saturdays}$. We use phonemes, syllables, and stresses for phonetic constraints such as rhyme or alliteration. Part of speech tags allow us to constrain word replacements to those of identical part of speech. Named entity tags allow us to constrain word replacements to those of the same word family. Lemmas allow us to equally transform all words with the same base, rather than all words of this one specific form.

sonant phoneme(s) preceding the nucleus. The coda is the consonant phoneme(s) following the nucleus. Both the onset and coda may be empty.

Our syllable parser uses the fourteen phonotactic rules of English (Harley 2006) to syllabify sequences of phonemes. The algorithm works by:

1. labelling a word’s nuclei ν
2. prepending onset phonemes ω to each nucleus ν while no phonotactic rule is broken
3. appending remaining coda phonemes κ to each nucleus ν while no phonotactic rule is broken

We do not use a dictionary for syllabic data. Hence, certain words of non-English origin are occasionally syllabified incorrectly.

Rhyme Rhymes are evaluated at the phoneme level by making use of a word’s *stress tail*, defined as the nucleus and coda of the syllable with the greatest stress along with all following syllables. Our rhyme scorer works by:

1. extracting the stress tail from two words
2. aligning the stress tails’ syllables
3. aligning the onset, nucleus, and coda of each syllable
4. scoring each aligned phoneme pair
5. scoring each syllable pair based on phoneme scores
6. scoring the word pair based on (stress tail) syllable scores

Phonemic scoring is computed one of two ways, depending on whether the phonemes are consonants or vowels. Consonant phonemes are characterized by three attributes: 1) voicing v , 2) place of articulation p , and 3) manner of articulation m . *Voicing* refers to whether vocal chords are used to pronounce a phoneme and may be represented as a Boolean value. *Place of articulation* refers to the point of contact where an obstruction occurs in the vocal tract to produce a consonant phoneme and can take one of seven nominal values: *alveolar*, *palatal*, *bilabial*, *velar*, *labiodental*, *interdental*, *glottal*. Finally, *manner of articulation* refers to the configuration and interaction of the tongue, lips, and palate when forming a consonant phoneme and can take one of seven nominal values: *fricative*, *stop*, *nasal*, *affricate*, *liquid*, *semivowel*, *aspirate*. Two consonant phonemes c_1 and c_2 are scored according to how well these three attributes match:

$$R_c(c_1, c_2) = \beta_v \delta_{v_1 v_2} + \beta_p \delta_{p_1 p_2} + \beta_m \delta_{m_1 m_2} \quad (7)$$

where δ_{ij} is the Kronecker delta function. We use the weights $\beta_v = 0.1$, $\beta_p = 0.45$, and $\beta_m = 0.45$

Vowel phonemes are similarly defined by three attributes: 1) voicing v , 2) frontness f , and 3) height h . *Frontness* refers to the distance from the back of the mouth when a vowel phoneme is formed. *Height* refers to the height of the tongue when a vowel phoneme is formed. Two vowel consonants u_1 and u_2 are scored according to (normalized) Euclidian distance between points of the English IPA vowel chart (Association 1999) (see Figure 2) superposed on the

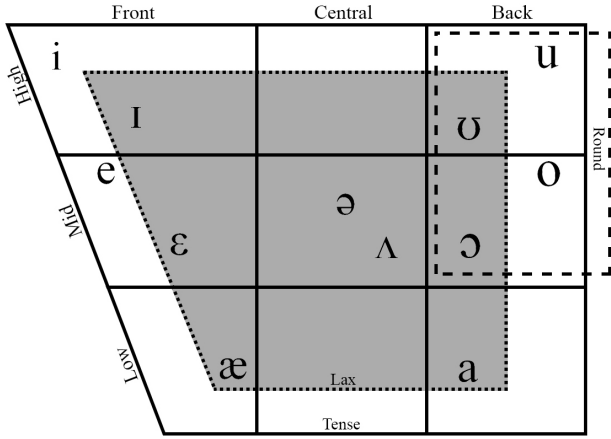


Figure 2: For rhyme scoring purposes, we estimate vowel similarity by finding the distance between phonemes on the standard IPA English Vowel Chart (1999). We superpose this chart on the Cartesian plane, with the horizontal axis measuring frontness and the vertical axis giving height.

Cartesian plane with axes frontness and height (voicing is ignored):

$$R_v(u_1, u_2) = \frac{1}{z} \sqrt{|f_1 - f_2|^2 + |h_1 - h_2|^2} \quad (8)$$

where z is a normalizing constant.

The equation for obtaining the rhyme score R_s of two syllables s_1 and s_2 is

$$R_s(s_1, s_2) = \beta_\omega \bar{R}_c(\omega_1, \omega_2) + \beta_\nu R_v(\nu_1, \nu_2) + \beta_\kappa \bar{R}_c(\kappa_1, \kappa_2) \quad (9)$$

where \bar{R}_c means the average consonant phoneme score, in case the onset or coda contain multiple (consonant) phonemes. To amplify the relative importance of the nuclei, we use the weights $\beta_\omega = \beta_\kappa = 0.125$, and $\beta_\nu = 0.75$.

Finally, the equation for obtaining the rhyme score R for words w_1 and w_2 , whose stress tails contain n_1 and n_2 syllables, is

$$R(w_1, w_2) = \frac{1}{n} \sum_{j=1}^n R_s(s_{1j}, s_{2j}) \quad (10)$$

where n is the greater of n_1 and n_2 , and syllable s_{ij} is the j th syllable in the stress tail of word i . To handle the case that $n_1 \neq n_2$, we define $R_s(s_1, -) = R_s(-, s_2) = 0$. Table 3 provides an example of this rhyme scoring process.

Output

Textual transformation can be used for various purposes, including to:

- alter a text’s sentiment
- alter a text’s meter
- introduce alliteration into a text

	stress tail				
w_1	W	IY	P	IY	NG
w_2	N	IY	D	IY	NG
Phoneme rhyme scores	n/a	1.0	0.45	1.0	1.0
Syllable rhyme scores			0.93		1.0
Word rhyme score					0.97

Table 3: Example of rhyme scoring between the stress tails of “weeping” and “needing”. This rhyme is imperfect (rhyme score < 1) due to the phoneme substitution of ‘D’ for ‘P’. Since one is voiced and the other is not, they do not share the same place of articulation but they do have the same manner of articulation, their (consonant) rhyme score is 0.45. In our rhyme algorithm, we exclude initial phonemes that are not part of the stress tail from alignment and scoring.

- alter a text’s rhyme scheme
- alter a text’s rhymes while preserving its rhyme scheme

Inspecting actual output from Lyrist is useful in understanding the transformation process. Some helpful indicators of success in transformed creative artefacts are:

- Adherence to intentions $\mathcal{I} = \langle \mathcal{I}_t, \mathcal{I}_c, \mathcal{I}_s \rangle$
- Adherence to constraints $\mathcal{C} = \langle \mathcal{C}_m, \mathcal{C}_r \rangle$
- Syntactic correctness (grammar)
- Semantic cohesion (beyond syntactic correctness, does the text carry meaning and seem human-written?)
- Obfuscation of identity of source

As an example, given the following text,

Sorrow’s my body on the waves
Sorrow’s a girl inside my cake
I live in a city sorrow built
It’s in my honey, it’s in my milk²

and the marking constraints $\mathcal{C}_m = \{P^m\}$, where

$$P^m(w) = \begin{cases} \text{TRUE} & \text{noun}(w) \vee \text{verb}(w) \vee \text{adj}(w) \\ \text{FALSE} & \text{otherwise} \end{cases}$$

the following (bold) words are marked for replacement:

*Sorrow’s my **body** on the waves*
*Sorrow’s a **girl** inside my **cake***
*I live in a **city** **sorrow** built*
*It’s in my **honey**, it’s in my **milk***

Then, given the thematic intention

$$\mathcal{I}_t = \langle w_{t_o} = \text{“sorrow”}, w_{t_n} = \text{“honor”} \rangle$$

and the structural intention

$$\mathcal{I}_s = \langle R_S = AAB B \rangle \text{ and the replacement constraints}$$

²from *Sorrow* by The National

- $$C_r = \{$$
1. spelling \neq original spelling
 2. lemma is not to equal to any other new lemma in the transformation
 3. part of speech is identical to original's part of speech
 4. rhyme score of matched line ending words is 1
 5. syllable count of original and replacement word is equal
- $$\}$$

the transformation produces the following:

*Honor's my heart on the crests
Honor's a woman inside my zest
I bide in a land honor divined
It's in my nectar, it's in my wine*

However, if the structural intention were then changed to

$I_s = \langle R_S = ABAB \rangle$
the result becomes

*Honor's my heart on the tides
Honor's a woman inside my roast
I bide in a land honor divined
It's in my nectar, it's in my toast*

Sentiment is easily modified through transformation. Consider the following source text:

*I loved beating these two terrible human beings.
I would never recommend that anyone use her lawyer, he is a total loser.*³

With the following intentions and constraints,

$I_t = \langle w_{t_o} = \text{"anger"}, w_{t_n} = \text{"kindness"} \rangle$
 $C_m = \{ \sigma(w, w_{t_o}) > 0.5, (\text{noun}(w) \vee \text{verb}(w) \vee \text{adj}(w)) \}$
this tweet's theme is changed from "anger" to "kindness". Only words similar to "anger" are marked, guaranteeing that the angriest words will be replaced and leaving more neutral words like "human" as they are:

I hated beating these two profound human beings. I would never refuse that anyone use her lawyer, he is a total sweetheart.

This transformation adheres to its intentions and constraints, retains syntactic correctness, and is semantically cohesive. The new text's overall tone and meaning, though awkward in places, has a distinctly kinder feel.

Alliteration is a poetic device that is simply achieved through transformation. Given the following text,

³Trump, Donald (@realDonaldTrump). 23 May 2013, 4:24 p.m. Tweet.

*Now folds the lily all her sweetness up*⁴

these intentions and constraints

$I_t = \langle w_{t_o} = \text{"nature"}, w_{t_n} = \text{"desert"} \rangle$

$I_s = \langle \text{alliterate} \rangle$

change the theme of this line of poetry from "nature" to "desert", while constraining for maximum alliteration:

Currently cracks the cactus all her creaminess up

Again, this transformation adheres to its intentions and constraints, retains syntactic correctness, and is semantically cohesive. The words "cracks" and "cactus" have distinct associations with "desert", and 50% of words match each other alliteratively.

Texts using imagery and/or stream-of-consciousness may be successfully transformed with relatively few constraints. Given this source text:

*Let us go then, you and I,
When the evening is spread out against the sky
Like a patient etherized upon a table;
Let us go, through certain half-deserted streets,
The muttering retreats
Of restless nights in one-night cheap hotels
And sawdust restaurants with oyster-shells:
Streets that follow like a tedious argument
Of insidious intent
To lead you to an overwhelming question...
Oh, do not ask, "What is it?"
Let us go and make our visit.*⁵

the following text is the result of a transformation with the thematic intention to transform from love to mystery ($I_t = \langle w_{t_o} = \text{"love"}, w_{t_n} = \text{"mystery"} \rangle$):

*Watch us proceed then, you and I,
When the morning is transmitted out against the horizon
Like a glaucoma trapped upon a riddle;
Watch us proceed, through unclear quarter shrouded alleyways,
The mumbling mysteries
Of unconvinced evenings in one-evening inexpensive resorts
And styrofoam eateries with mussel-bomblets:
Alleyways that after like a laborious question
Of pernicious intention
To advantage you to a vast conundrum...
Oh, do not determine, "what is it?"
Watch us proceed and disentangle our trip.*

⁴Excerpt from *Summer Night* by Alfred Tennyson

⁵First stanza of *The Love Song of J. Alfred Prufrock* by T. S. Eliot

The primary flaw of this transformation is occasional syntactic incorrectness. The eighth line, where “follow” is replaced with “after”, is one such example. Errors such as this are preventable only with more powerful and more specific part of speech parsers. Part of speech code sets such as Penn Treebank are quite general and do not include data such as transitivity for verbs, number for pronouns, etc.

As a final example consider the following song lyrics, transformed by finding similar rhyme words with no overarching theme while maintaining rhyme scheme:

*Lights go out and I can't be saved
Tides that I tried to swim against
Have brought me down upon my knees
Oh I beg, I beg and plead
Come out of the things unsaid
Shoot an apple off my head
And a trouble that can't be named
The tiger's waiting to be tamed⁶*

***Bulbs run out and I can't be hauled
Currents that I attempted to run enthralled
Have drawn me down upon my thighs
Oh I sue, I sue and prize
Come out of ways paternal
Launch an orange off my colonel
And a danger that can't be convened
The tigress's awaiting to be weaned***

Discussion

In a survey conducted for preliminary evaluation of Lyrst, thirty-nine participants responded to questions pertaining to artefact 1) quality, 2) theme, and 3) source identification.

To measure quality, participants rated four textual artefacts using a Likert scale ranging from 1 to 5. The mean score was 3.05, with a standard deviation of 0.24, suggesting mainly neutral reactions to the transformed texts. One possible explanation is that “overall quality” is too general a measurement of success for creative artefacts. In the future we plan to have respondents evaluate rhyme schemes, rhyme choice, imagery associated with new theme, etc. according to a well-defined rubric for each of these categories.

To measure thematic accuracy, we transformed one original piece six times, each time using a distinct emotion as the new theme $w_{t,n}$: “excitement”, “dignity”, “love”, “confusion”, “paranoia”, and “sadness”. These were arbitrarily chosen and could have been replaced by other single-word themes. Details of the transformation \mathcal{T} were kept private from participants. For each transformed piece, participants selected one emotion from the six as the emotion it best evoked. Table 4 gives the results in a confusion matrix. Overall the correct emotion was identified over two-thirds of the time. This suggests that drawing analogies from word embeddings is a good way to reflect thematic intention \mathcal{I}_t in textual artefacts. Indeed, the overall correctness result

⁶*Clocks* by Coldplay

	E	D	L	C	P	S
Excitement	.904	0	.048	.048	0	0
Dignity	.05	.9	0	0	0	.05
Love	.444	0	.5	0	0	.056
Confusion	0	0	.368	.421	.053	.158
Paranoia	0	.05	0	.35	.55	.05
Sadness	.053	.053	0	0	0	.894

Table 4: Sentiment confusion matrix for correct sentiment identification by survey participants. Certain sentiment pairs, such as dignity and confusion, were never confounded. But other sentiments, such as love and excitement, were confounded frequently. Column sentiments are abbreviated for legibility. The correct sentiment guess rate is bolded along the diagonal.

was not higher because “love” was confounded with “excitement”, and “confusion” with “love”. It can be argued that the emotions “love” and “excitement” share many characteristics; this can be similarly argued for “confusion” and “love”. Such confounding of similar themes is to be expected to some extent and may even be considered as evidence for the semantic quality of the language model \mathcal{L} .

Participants were shown transformations of four popular English songs and asked to guess the title or artist of the original text. Afterwards, participants indicated which pop songs from a larger list they were familiar with, including the four original songs. Figure 3 shows the results. Guesses were counted as incorrect if the participant admitted to being familiar with a song but was unable to identify it after transformation. Overall two-thirds of guesses were incorrect, suggesting that the identity of even familiar texts is obfuscated after transformation. We see this as generally advantageous; a CC process is more likely to be deemed creative by unbiased observers if its technical specifics are not apparent.

Of course, even though this process guarantees accurate usage of constraints and mapping from original to new texts, it necessarily bears the combined error rates of its third-party dependencies, such as part of speech parsers. These errors can cause loss of syntactic correctness and semantic cohesion. After cleaning one’s data, this condition can only be improved by improving the third-party tools themselves.

Additionally, an inherent flaw of string-based word embedding is relating words of identical spellings with different meanings, such as the noun and verb both spelled “wind”. Any word with multiple definitions is probably less-than-ideally represented in such models.

Conclusion

Text transformation via constraints and word embedding is simple in theory, and powerful in practice. This concept builds upon research in computational linguistics and CC. Textual transformation requires large amounts of textual data for its language model and may additionally benefit from a large pool of original texts to transform. Word embedding allows computer systems to estimate word meaning and suggest analogous word replacements. Prioritized

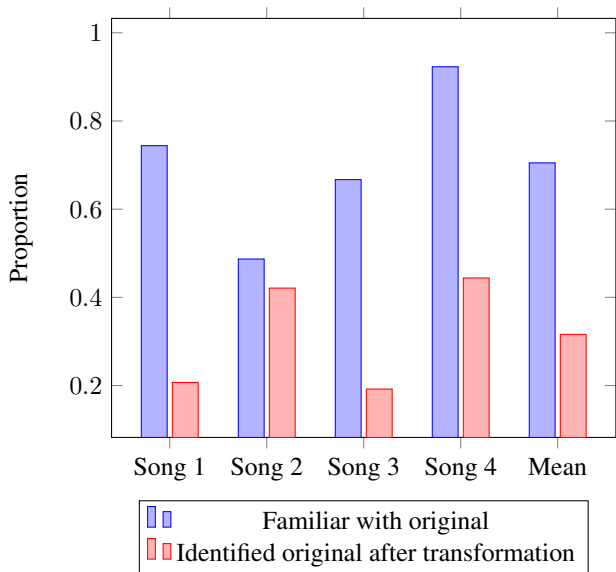


Figure 3: Proportion of participants that correctly identified original pop songs after textual transformation. Though most participants were familiar with the original text of four famous pop songs, most were unable to identify them from their transformed counterparts. This suggests that text transformation with Lyrist obfuscates the identity of texts well.

constraints deterministically filter out unwanted replacement words. A mechanism for representing intention provides a motivating goal, and the potential for inspiration driving this intention admits the possibility of greater system autonomy in the future. The process obfuscates textual origin and produces generally cohesive, thematically accurate text.

In the future, we plan to build an interactive web tool to showcase creative intentions and this transformation method. We also plan to construct a cultural database that allows the creation of artefacts incorporating cultural intention \mathcal{I}_c . We will explore pairing Lyrist with a system that uses an English grammar to break original texts into sentences, clauses, and phrases, allowing text transformation to function independently of complete human-written texts. Lyrist is currently being integrated with a musical artefact generator (Bodily, Bay, and Ventura 2017). We will explore the results of this pairing in future work.

References

Association, I. P. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press.

Bateman, J., and Zock, M. 2003. Natural language generation. In *The Oxford Handbook of Computational Linguistics 2nd edition*. Oxford University Press.

Bodily, P.; Bay, B.; and Ventura, D. 2017. Human-level concept learning in computational creativity. In *Proceedings of the Eighth International Conference on Computational Creativity*, to appear.

Brownstein, J.; Yangarber, R.; and Astagneau, P. 2013. Algodan publications 2008-2013. *Journal of Intelligent Information Systems* 1–19.

Colton, S.; Goodwin, J.; and Veale, T. 2012. Full FACE poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.

Davies, M. 2008. *The Corpus of Contemporary American English*. BYU, Brigham Young University.

Davies, M. 2016. The Corpus of News on the Web.

Gervás, P.; Hervás, R.; and Robinson, J. R. 2007. Difficulties and challenges in automatic poem generation: Five years of research at UCM. *e-poetry*.

Gervás, P. 2002. Exploring quantitative evaluations of the creativity of automatic poets. In *Proceedings of the Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence*.

Gonçalo Oliveira, H. R.; Cardoso, F. A.; and Pereira, F. C. 2007. Tra-la-lyrics: An approach to generate text based on rhythm. In Cardoso, A., and Wiggins, G., eds., *Proceedings of the 4th International Joint Workshop on Computational Creativity*.

Gonçalo Oliveira, H. 2015. Tra-la-lyrics 2.0: Automatic generation of song lyrics on a semantic domain. *Journal of Artificial General Intelligence* 6(1):87–110.

Harley, H. 2006. *English Words: A Linguistic Introduction*. Blackwell Publishing Ltd.

Hirjee, H., and Brown, D. G. 2009. Automatic detection of internal and imperfect rhymes in rap lyrics. In *10th International Society for Music Information Retrieval Conference*, 711–716.

Kominek, J., and Black, A. W. 2004. The CMU arctic speech databases. In *Proceedings of the Fifth ISCA Workshop on Speech Synthesis*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, 55–60.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19(2):313–330.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Monteith, K.; Martinez, T.; and Ventura, D. 2012. Automatic generation of melodic accompaniments for lyrics. In *Proceedings of the Third International Conference on Computational Creativity*, 87–94.

Tobing, B. C., and Manurung, R. 2015. A chart generation system for topical metrical poetry. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 308–314.

Computational Creativity via Human-Level Concept Learning

Paul Bodily, Benjamin Bay, and Dan Ventura

Computer Science Department
Brigham Young University
Provo, UT 84602 USA

paulmbodily@gmail.com, benjamin.bay@gmail.com, ventura@cs.byu.edu

Abstract

A common framework is helpful for effective evaluation, collaboration, and incremental development of creative systems. The Hierarchical Bayesian Program Learning (HBPL) framework was recently shown to be highly effective at learning human-level concepts, achieving new standards of performance in one-shot classification, parsing, and generation of hand-written characters. We argue that the HBPL framework is well-suited for modeling creative artefacts in general, one reason being that it allows explicit modeling of intention, structure, and substructure. Furthermore, the major challenge to the HBPL framework, namely how joint distributions should be factored, focuses system designers' attention on the philosophical debates that occur among artists themselves, suggesting that the HBPL framework might also serve as a more precise scaffolding for such debates. We demonstrate the framework's efficacy using lyrical composition as a specific example. In addition to being able to generate novel artefacts, we illustrate how HBPL models can be used to incorporate creative knowledge in broader applications including recommendation systems.

Introduction

People possess the ability to learn and combine concepts they already know to understand and even create new concepts. As an example, many pedagogical models (e.g., (Englemann and Bruner 1974)) teach children to read by systematically mastering and combining simple concepts: symbols represent sounds; symbols are read left to right; sounds are combined to form words; periods delimit phrases; sentences wrap to subsequent lines, etc. This process of hierarchical learning is at the heart of a branch of machine learning called *human-level concept learning*. Human-level concept learning is characterized by three fundamental ideas (Lake, Salakhutdinov, and Tenenbaum 2015):

- *Compositionality* - observations are constructed through a combination of parts
- *Causality* - capturing abstract representations of the causal process that produces an artefact
- *Learning-to-learn* - parameters, constraints, parts, etc. are learned from training with related concepts and then applied to learning novel concepts

Hierarchical Bayesian program learning (HBPL) describes a framework that models human-level concept learning. This framework has recently been shown to be extremely effective (better even than deep-learning algorithms) in one-shot classification, parsing, and generation of hand-written characters (Lake, Salakhutdinov, and Tenenbaum 2015). The HBPL model for hand-written characters works by factoring a joint probability distribution over characters ψ into a product of conditional distributions,

$$P(\psi) = P(\kappa) \prod_{i=1}^{\kappa} P(n_i|\kappa)P(S_i|i, n_i)P(R_i|S_1, \dots, S_{i-1}), \quad (1)$$

where each conditional distribution is a model of a *sub-concept*: $P(\kappa)$ models the number of strokes per character; $P(n_i|\kappa)$ models the number of sub-strokes for the i th stroke for a character with κ strokes; $P(S_i|i, n_i)$ models the i th stroke with n_i sub-strokes; and $P(R_i|S_1, \dots, S_{i-1})$ models the relation of the i th stroke to the previous strokes. Some of these models are further decomposed. This process of decomposition allows the system to empirically learn sub-concepts in order to learn and generate new character types.

In this paper we investigate concept learning as a tool for building computationally creative systems. In particular, we find that the HBPL model provides a powerful framework for producing novel, typical artefacts that include elements of surprise by virtue of its wide range of expression.

As a proof of concept, we demonstrate the application of the HBPL model to the problem of lyrical pop music composition; however, the principles are readily applicable in other domains. Lyrical pop music is an ideal subject insofar as it naturally decomposes into multiple subconcepts, each of which can be further factored. The system we describe also demonstrates how existing models can be incorporated in defining subconcept distributions, using the specific example of Pachet et al.'s constrained Markov model (2011).

Modeling with HBPL

The most significant challenge to the HBPL model is deciding how and how far to factor the joint distribution. Bayes' theorem suggests that the factoring is irrelevant: any factoring should reproduce the joint when terms are multiplied:

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A).$$

However, in practice we are only ever able to approximate distributions. Furthermore we at times make unproven in-

dependence assumptions to increase the power of our models (as discussed below). The factorization therefore leaves some “fingerprints” on the artefacts it produces according to the extent that each of the factors is accurately modeled.

Given that the space of possible artefacts is essentially infinite for many domains, it can be challenging to accurately train models for each subconcept given the relatively few artefacts that have actually been created. But often an approximation is sufficient to get a reasonable, working model. That we must use approximate distributions encourages the use of a modular framework for a few reasons. First, a modular framework affords the metacreator the opportunity to improve upon or substitute alternative approximative distributions for subcomponents. Second, multiple approximations can be combined to create improved approximations.

Depending on the complexity of the artefact class, the decision of how to factor the joint distribution can have significant impact on the power of the model. Some factorings generate subconcept models that may be easier to approximate. Some factorings may lend themselves to more reasonable independence assumptions. Choosing a good factorization often requires a deep understanding of the artefact domain.

For relatively simple artefacts, the decision of how to factor the joint is more straightforward. For example, consider just a few of the independence assumptions that Lake et al.’s model makes about hand-written characters (2015):

1. The number of substrokes per stroke, though dependent on the number of strokes, is independent from the number of substrokes in previous strokes and from the stroke-order position of the current stroke.
2. A substroke identity (i.e., shape) depends on the stroke-order position and the number of substrokes in the current stroke, but not directly on the total number of strokes in the character nor on the substroke identities of any but the directly previous substroke.
3. How strokes connect to previous strokes is independent of the number of strokes, substrokes, or substroke identities.

Initially these all seem like very reasonable simplifying assumptions, especially when considering how well the model performs. However if hand-written characters were more widely considered and utilized as an art-form, there may be some disagreement about how accurate these assumptions really are. Furthermore, the greater disagreements would likely come from what this choice of factoring says about the intuition behind how a character is generated: first randomly select a number of strokes κ ; then select a number of substrokes n for each of those strokes based on κ ; select the substroke shapes based on n and κ ; and finally select the relationship between strokes. For most non-artistic character implementers, there is nothing wrong with this intuition. However, a calligrapher might feel that generating a new character really starts with choosing a substroke shape or a relationship between strokes. Note that the HBPL model could easily be adapted to model either of these alternative intuitions; but more importantly it highlights the debate of whether or not it is important *what* the model is doing as long as it appropriately classifies and generates character types.

In contrast, consider some potential independence assumptions and intuition represented in a model of lyrical compositions:

1. The structure, harmony, melody and lyrics are all independent of the inspiring source, given the intention.
2. The pitches of the melody are dependent on the harmony.
3. The number of syllables in the lyrics are dependent on the number of notes in the melody.
4. The lyrics are independent of the harmony, given the melody.

There are likely to be disagreements over some aspects of this factorization, reflecting philosophical biases of individual artists. Similar debates would arise, for example, in asking song-writers, “which do you write first: the lyrics or the melody?” Or asking story-writers, “which comes first: the characters or the story?” The fact remains that the same artefacts are producible by multiple factorizations and the majority of those who appreciate the creativity of a song or a story do so without any knowledge of which factorization created it. These debates about how the model should be factored are the very same debates in which artists themselves engaged. By requiring the metacreator to precisely define how the joint should be factored, the HBPL model focuses attention on these debates and represents a computational framework in which differing perspectives can be readily compared and evaluated. For a discussion of different philosophies of lyrical composition and how they are represented as factorizations of the joint distribution over lyrical compositions see Bodily and Ventura (2017).

Composition

Analogous to equation 1, we define the conditional distribution on compositions γ , given an inspiration ι , as follows,

$$P(\gamma|\iota) = P(\nu|\iota)P(\tau|\nu)P(\eta|\nu, \tau)P(\mu|\nu, \tau, \eta)P(\lambda|\nu, \tau, \mu),$$

with the following definitions:

$$\begin{aligned} P(\nu|\iota) &= \text{distribution over intentions } \nu \text{ given } \iota, \\ P(\tau|\nu) &= \text{distribution over structure } \tau \text{ given } \nu, \\ P(\eta|\nu, \tau) &= \text{distribution over harmony } \eta \text{ given } \nu \text{ and } \tau, \\ P(\mu|\nu, \tau, \eta) &= \text{distribution over melody } \mu \text{ given } \nu, \tau, \text{ and } \eta, \text{ and} \\ P(\lambda|\nu, \tau, \mu) &= \text{distribution over lyrics } \lambda \text{ given } \nu, \tau, \text{ and } \mu. \end{aligned}$$

Although this factorization is dependent on the domain of lyrical composition, there are strong cross-domain parallels for many of the factors, which we will examine. This factorization of the distribution over compositions makes several independence assumptions which are discussed by Bodily and Ventura (2017). Given our factorization decisions, we generally find that the learned distributions broadly agree with musical intuition about how each of the subconcepts is defined as discussed in figure captions.

Intention, $P(\nu|\iota)$ *Intention* can be defined as the objectives which influence the creation of an artefact and can address several different facets (Bay, Bodily, and Ventura 2017):

- *Thematic intention* - the semantic purpose of the artefact (e.g., subject, emotion)
- *Cultural intention* - the sociocultural context for the artefact (e.g., society, language, era, genre)
- *Structural intention* - the target organization or arrangement of an artefact (e.g., technique, rhyme scheme, meter)

Whereas intention ν represents *what/how* we want to communicate, the *inspiration* ι represents the inspiring source for ν or *why* we want to communicate ν . Although many creative systems model intention (e.g., via a fixed intention, a user-defined intention, or randomly selecting an intention), a major advantage to the HBPL model is that we can explicitly condition the intention for an artefact on an inspiration. We discuss inspiration more below.

In our working lyrical composition example, we use a randomly selected thematic intention. Though several of the remaining subconcept models are conditioned on ν , it is only explicitly discussed in relation to $P(\lambda|\nu, \tau, \mu)$. We include it elsewhere as a reminder that intention can and should influence creativity wherever possible. We will assume that conditioning on ν is elsewhere accomplished by conditioning training on data representative of ν and leave a deeper exploration of its implementation for future work.

Structure, $P(\tau|\nu)$ In many domains of creativity structure can be thought of hierarchically. For example in a computer game the global structure may describe aspects of the flow between levels, but the levels themselves also have significant substructural elements that are intuitively independent from the global structure. We can thus factor our model of structure τ as

$$P(\tau|\nu) = P(\zeta|\nu)P(\sigma|\nu, \zeta)$$

where

$P(\zeta|\nu)$ = distribution over global structure ζ given ν and

$P(\sigma|\nu, \zeta)$ = distribution over segment structure σ given ν and ζ .

Global structure defines the boundary and relationships between subparts of an artefact. Examples might include the abstract sequence of plot line elements in story writing (e.g., “hero cycle” vs “tragedy”) or the proportions of different abstract food groups in recipe generation (e.g., “chili” vs “sandwich”) (e.g., (Morris et al. 2012)). In lyrical pop music, these subparts are readily apparent in the sequence of verses (V) and choruses (C) (which define large-scale repetitions in one or more musical viewpoints) and intros (I), outros (O), and bridges (B) (generally not wholly repeated). We refer to a subpart in our model as a *segment* and its value (e.g., “verse”) as its *segment type*. A global structure for lyrical composition is a sequence of segment types $\zeta = (\zeta_1, \dots, \zeta_n)$ with arbitrary length, where $\zeta_i \in \{I, V, C, B, O\}$. We define $|\zeta|$ as the number of segment types in ζ .

There are several ways to approximate $P(\zeta|\nu)$. One severely limited approximation is a *fixed* structure (e.g., I,V,C,V,C,B,C,O). Despite the range of possible compositions that are uncomputable by this approximation, this lim-

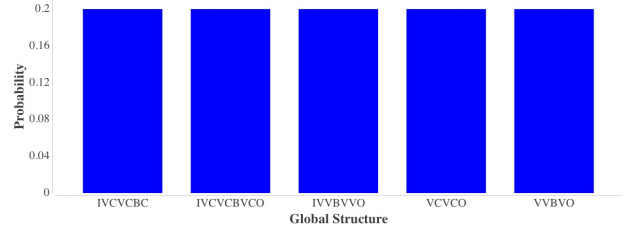


Figure 1: A visual representation of a possible probability distribution over global song structures composed of verses (V), choruses (C), intros (I), outros (O), and bridges (B).

itation would likely be overlooked if enough variation exists in other subcomponent models.

A second approximation is a *distributional model* which learns a multinomial distribution of possible structures from a corpus of composition artefacts (e.g., see Figure 1). The disadvantage to the distributional model is that it can only produce structures seen in training.

A third, more powerful approximation uses a *constrained Markov model*. This model factors $P(\zeta|\nu)$ into a distribution over the number of segments in a song, $P(|\zeta|)$, and a single-order Markov model for sequences of segment types:

$$P(\zeta|\nu) = P(|\zeta|)P(\zeta_1) \prod_{i=2}^{|\zeta|} P(\zeta_i|\zeta_{i-1})$$

Note that an unconstrained, unsmoothed Markov model for $P(\zeta_i|\zeta_{i-1})$ provides no guarantee that a sequence of length $|\zeta|$ can or will be generated, nor that the sequence will end naturally (e.g., with an outro). With Pachet et al.’s *constrained* Markov model we can constrain the length and the way the sequence ends. This modifies the way $P(\zeta|\nu)$ is factored by conditioning ζ_i on both i and ζ_{i-1} :

$$P(\zeta|\nu) = P(|\zeta|)P(\zeta_1) \prod_{i=2}^{|\zeta|} P(\zeta_i|i, \zeta_{i-1})$$

When generating, a length is sampled from $P(|\zeta|)$ and a constrained Markov model for the sampled length is constructed from the unconstrained model $P(\zeta_i|\zeta_{i-1})$ with the added constraint that the song must end on an “end” token. This model is capable of creating sensible structures of reasonable length that were not seen in the training data. Empirical distributions for approximating $P(|\zeta|)$ and $P(\zeta_i|\zeta_{i-1})$ are shown in Figures 2 and 3 respectively.

A fourth possible solution for generating global structure would be to use a generative grammar, learned or manually constructed, similar to what was done by (Steedman 1984).

In addition to global structure, we also model *segment structure*, $P(\sigma|\nu, \zeta)$. Though this segment structure could be included as part of global structure, modeling this substructure independently leverages principles of abstraction and polymorphism in order to facilitate novel combinations of substructures. For example in story-generation the global structure might dictate something about the abstract content of each paragraph (e.g., protagonist faces a trial, protagonist

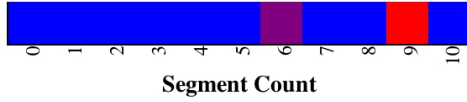


Figure 2: A visual representation of a possible probability distribution over the number of segments per song. Red corresponds to high probability, blue to low.

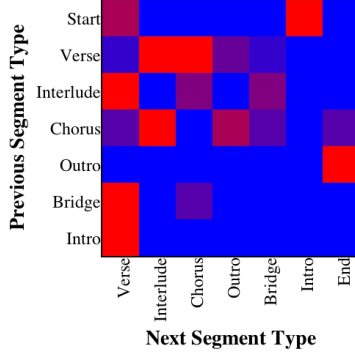


Figure 3: A visual representation of a possible single-order Markov transition matrix for segment types. Red corresponds to high probability, blue to low. The results largely agree with intuition. For example, songs generally start with an intro and occasionally with a verse; songs generally end with an outro and occasionally a chorus; and segments of the same type do not generally follow one another.

learns lesson, etc.), whereas the segment structure might define the narrative style for the paragraph (e.g., dramatic visualization, retrospection, dialogue, etc.) or add definition to the abstract content (e.g., the trial is a storm, the trial is losing a loved one, etc.). Modeling these structures independently enables the model to combine narrative styles with plot elements in ways that were not seen during training.

A segment in a composition (e.g., a verse) exhibits structure in the number of measures, the number of syllables or notes per segment, which lyrics rhyme or repeat, and patterns in harmony, pitch, or rhythm. We define a segment structure for lyrical composition as a sequence of pairs $\sigma = ((l_1, C_1), \dots, (l_{|\zeta|}, C_{|\zeta|}))$, where l_i is the measure length of the i th segment (corresponding to ζ_i) and $C_i = \{c_{i1}, \dots, c_{in}\}$ is a set of constraints which apply to the i th segment.

Constraints define restrictions on different musical viewpoints in order to create rhyme and repetitive motifs. A constraint, c_{ij} , is defined for a particular viewpoint $v \in \{Harmony, Pitch, Rhythm, Lyric\}$; with a condition $d \in \{Equals, Matches, RhymesWith, HasExpectation\}$; with a Boolean value t that defines whether the condition d needs to be satisfied or unsatisfied in order to satisfy the constraint c_{ij} ; and with $m \in [0, l_i]$ and $b \in [0.0, bpm_m)$ representing the measure and beat offset within the segment to which the constraint applies (bpm_m is the beats per measure of m). Each condition d has different sub-variables

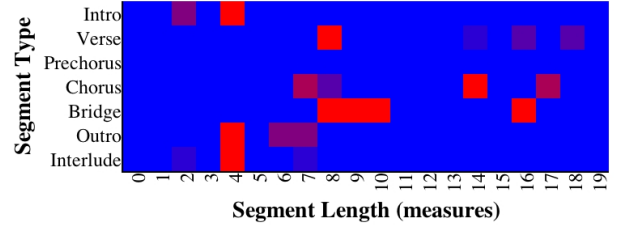


Figure 4: A visual representation of an empirically derived probability distribution over song segment lengths, conditioned on segment type. Red corresponds to high probability, blue to low. The results largely agree with intuition: intros, outros, and interludes tend to be shorter; verses, bridges and choruses tend to be longer.

and dimensionality:

- *Equals* conditions - $c_{ij} = (v, d = Equals, t, m, b, S)$, where to satisfy d , the v token at or near measure m , beat b must equal a v token in the set of tokens S if t is *true* and must not equal any v token in S if t is *false*.
- *Matches* conditions - $c_{ij} = (v, d = Matches, t, m, b, m_2, b_2)$, where to satisfy d the v token at or near measure m , beat b and at or near measure m_2 , beat b_2 within the segment must be equal if t is *true* and not equal if t is *false*.
- *RhymesWith* conditions - $c_{ij} = (v = Lyric, d = RhymesWith, t, m, b, m_2, b_2)$, where to satisfy d the *Lyric* tokens at or near measure m , beat b and at or near measure m_2 , beat b_2 within the segment must rhyme if t is *true* and not rhyme if t is *false*.
- *HasExpectation* conditions - $c_{ij} = (v, d = HasExpectation, t, m, b, s)$, where to satisfy d the v token at or near measure m , beat b must have an expectation value above a threshold s if t is *true* and not have an expectation value above s if t is *false*. This constraint can be used to create a structure of expectation (as discussed by Meyer (2008)) in order to model patterns of surprise and tension.

Note that the attribute t could allow the system to learn how to intelligently break rules. For example, the system could intelligently learn when *not* to rhyme when perhaps a rhyme would normally be expected.

We define the distribution over segment structures σ as

$$P(\sigma|\nu, \zeta) = \prod_{i=1}^{|\zeta|} P(C_i|l_i)P(l_i|\zeta_i).$$

To approximate $P(l_i|\zeta_i)$ we can learn a probability distribution over segment lengths conditioned on segment type (see Figure 4). Under the assumption that the constraint set for a segment is independent of the segment type given its length, we can approximate $P(C_i|l_i)$ using a probability distribution over sets of constraints conditioned on segment length (e.g., see Figure 5).

Much of the work that has been done with finite-length Markov processes with constraints has required the user to

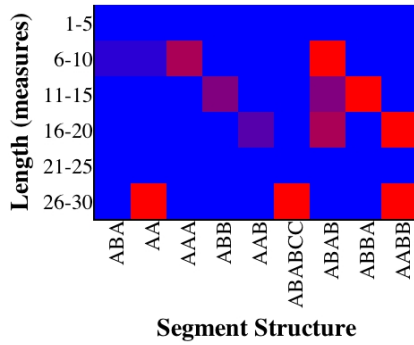


Figure 5: A visual representation of an empirically derived probability distribution over song segment rhyme structures conditioned on segment length. Red corresponds to high probability, blue to low.

specify the desired constraints in the composition process (e.g., (Pachet and Roy 2014; Barbieri et al. 2012)). This step of learning a model of constraints gives the system increased autonomy to choose its *own* constraints and then generate artefacts to meet those constraints.

With regard to modeling distributions for implicit features of an artifact (e.g., rhyme constraints), empirically-derived distributions can incur significant AI challenges. Artefacts used for training often fail to label global and even segment structure, and therefore these implicit features must be manually labeled or somehow inferred. Though our current system learns structure from a small manually-annotated dataset, our goal in future work is to use sequence alignment over multiple viewpoints to infer global structure, finding regions of a composition where harmony, melody, and lyrics all match (i.e., chorus) or where only harmony and melody match (i.e., verse). Sequence alignment is also a promising approach to finding segment structure (e.g., Hirjee and Brown (2010) use alignment to detect rhyme scheme).

Having modeled the abstract structural representation, the system proceeds to model the *operational* representation of the artefact (e.g., paint strokes, narrative text, recipe ingredients, etc.). Whether modeled jointly or factored, the operational variables describing the artefact composition are conditioned on the constraints imposed by the intention and global/segment structure. Adapting Pachet and Roy’s definition of a jazz leadsheet (2014), we define the operational representation of a lyrical composition as parallel sequences of chords η , notes μ , and lyrics λ each with the same total duration. η , μ , and λ are defined in the following sections.

Harmony, $P(\eta|\nu, \tau)$ We define a harmony as a sequence of positioned chords $\eta = (C_1, \dots, C_n)$ of arbitrary length. Each positioned chord $C_i = (I_i, d_i)$ has an identity $I_i = (r_i, q_i, s_i)$, with root pitch $r_i \in [0, 11]$, chord quality q_i (e.g., major, minor, dominant, etc.¹), and bass pitch $s_i \in [0, 11]$; and a duration $d \in \mathbb{R}_{>0}$. We normalize all root and bass

¹possible values for q_i are defined according to the MusicXML 2.0 specification for chord qualities

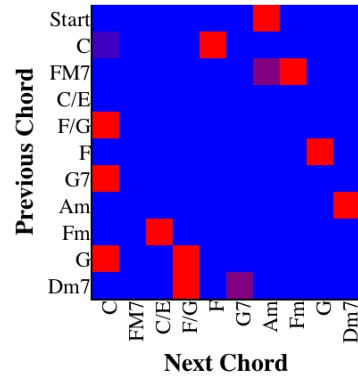


Figure 6: A subsection of a visual representation of an empirically derived single-order Markov transition matrix for harmonic chord sequences for chorus segments. Red corresponds to high probability, blue to low. As expected for songs normalized to the key of C major, there is high probability that the song starts on a C major chord.

itches based on the labeled key signature of the training instance at the harmony position.

We can factor $P(\eta|\nu, \tau)$ into independent sequential models regulating chord duration and chord identity:

$$P(\eta|\nu, \tau) = P(I_1|\tau)P(d_1|\tau) \prod_{i=2}^n P(I_i|I_{i-1}, \tau)P(d_i|d_1, \dots, d_{i-1}, \tau).$$

In this formulation, the length of the sequence n is dynamically determined such that $\sum_{i=0}^n d_i$ equals the segment duration.

Deciding how to implement $P(I_i|I_{i-1}, \tau)$ and $P(d_i|d_1, \dots, d_{i-1}, \tau)$ is non-trivial. A few possibilities for probabilistic sequence models include:

1. a *fixed generator* generates a fixed token, essentially ignoring conditioned variables
2. a *probability distribution* over tokens, conditioned on segment type and/or beat position, but not previous token
3. a *Markov model* that generates a new sequence for each segment, independent of segment type
4. a *set of Markov models* - one model per segment type
5. a *hidden Markov model* - hidden states representing the segment type

Each model has limitations that must be considered in the context for which it is intended. Of these, our implementation uses model 4 for $P(I_i|I_{i-1}, \tau)$ (see Figure 6) and model 2 for $P(d_i|d_1, \dots, d_{i-1}, \tau)$ (for a discussion of the relative musical merits of these models see Bodily and Ventura (2017)).

The decision to assume that duration and chord are independent, though potentially erroneous, is deliberate. This is based on the reasoning that the strength of a probabilistic model depends on the number of instances used to train the model. Each time a distribution adds a conditional variable,

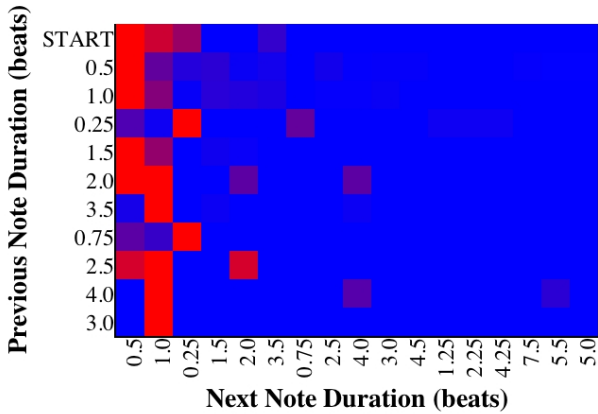


Figure 7: A visual representation of an empirically derived single-order Markov model for melodic rhythm durations for verse segments in 4/4. Red corresponds to high probability, blue to low.

the power of the model is reduced. We feel that the duration and chord are sufficiently independent that the model strength recovered by assuming independence outweighs the cost of ignoring any dependence between them.

Melody, $P(\mu|\nu, \tau, \eta)$ A melody is a sequence of positioned notes $\mu = (N_1, \dots, N_n)$ of arbitrary length. Each note $N_i = (p_i, d_i)$ has a pitch $p_i \in [-1, 127]$ (corresponding to a MIDI note value, -1 representing a rest) and a duration $d_i \in \mathbb{R}_{>0}$. We factor $P(\mu|\nu, \tau, \eta)$ into independent sequential models regulating note pitch and duration:

$$P(\mu|\nu, \tau, \eta) = P(p_1|\eta)P(d_1|\tau) \prod_{i=2}^n P(p_i|p_{i-1}, \eta)P(d_i|d_{i-1}, \tau).$$

The length of the sequence n is dynamically determined such that $\sum_{i=0}^n d_i$ does not exceed the segment duration.

Of these models only pitch is conditioned on η . To model $P(p_i|p_{i-1}, \eta)$ our implementation uses a single-order Markov chain of scale steps where the scale is defined by the contextual harmony of η . To model $P(d_i|d_{i-1}, \tau)$ we use a segment-specific Markov chain of note durations (see Figure 7). Any of the probabilistic sequence models considered for harmony could also be considered here.

Lyrics, $P(\lambda|\nu, \tau, \mu)$ Several models of natural language generation (NLG) and in particular NLG in poetry and music have been published (Paris, Swartout, and Mann 2013). As these models continue to improve, so will their application in lyrical composition. This demonstrates the robustness of the HBPL framework: as improved submodels are conceived and implemented, the joint model is also improved.

We define lyrics as a sequence of stressed syllables $\lambda = (S_1, \dots, S_n)$ where $|\lambda| \leq |\mu|$. A stressed syllable $S_i = (t_i, p_i, \epsilon_i)$ has a text representation t_i , a pronunciation p_i (e.g., sequence of ARPAbet phonemes), and a stress $\epsilon_i \in [0, 2]$. Each syllable $S_i \in \lambda$ corresponds to one and only one note $N_j \in \mu$.

We factor $P(\lambda|\nu, \tau, \mu)$ to construct λ as a sequence of lyric phrases (ϕ_1, \dots, ϕ_n) where the number of phrases n and

the length l_{ϕ_i} (in syllables) of each phrase are computed as a function of the notes in μ and the rhyme constraints in τ (i.e., we assume rhyme constraints denote phrase endings):

$$P(\lambda|\nu, \tau, \mu) = \prod_{i=1}^n P(\phi_i|l_{\phi_i}, \nu, \tau)P(l_{\phi_i}|\mu, \tau).$$

We empirically derive $P(l_{\phi_i}|\mu, \tau)$. For $P(\phi_i|l_{\phi_i}, \nu, \tau)$ we create a probability distribution of lyric templates conditioned on l_{ϕ_i} which we use to sample templates. These templates, the *RhymesWith* constraints of τ , and ν are given as input to an independent module that generates novel, intentioned lyrics (see Bay, Bodily, and Ventura (2017)). The module uses existing lyric segments as syntactic templates for the creation of novel lyric segments. It intelligently selects and replaces words based on 1) semantic similarity, 2) part-of-speech tag, 3) the cultural and thematic intention of ν , and 4) the rhyme constraints imposed by τ .

The advantage of using a template-based approach to lyrics generation is that it maintains syntactic coherence. The primary shortcomings are that resulting lyrics provide limited syntactic novelty from the training data and make no inherent effort at providing global semantic cohesion.

A Note on Constrained Markov Models Pachet et al.’s constrained Markov model requires that the length of the sequence be defined *a priori* (2011). One short-coming in our current implementation is that because we have included duration as part of the definition for both harmony and melody (rather than having each chord or note representative of a fixed duration as demonstrated by Pachet and Roy (2014)) the length of a harmony or melody sequence depends on the durations of each sampled chord or note. While this violates the Markov property and prevents us from being able to effectively use constrained Markov models, we favor the current implementation for reasons related to data sparsity issues and the complexity of implementing higher-order constrained (hidden) Markov model. We hope in the future to overcome both of these hurdles and to shift to “Markov-friendly” definitions for melody and harmony in order to more fully incorporate the constraints defined in τ using constrained Markov or constrained hidden Markov models.

Results and Discussion

We present results of implementing the HBPL framework in the context of a discussion of some of the model’s implications. We trained submodels on a small manually-annotated subset of the Wikifonia leadsheet dataset.

Using the Joint as a Submodel

Because of the hierarchical nature of HBPL, a joint model of an artefact class (e.g., the model of $P(\gamma|\iota)$ just described) can serve as a submodel for other models. For example, we define the joint probability distribution on inspirations ι , compositions γ , and renderings ρ^m as follows,

$$P(\iota, \gamma, \rho^1, \dots, \rho^m) = P(\iota)P(\gamma|\iota) \prod_{m=1}^M P(\rho^m|\iota, \gamma).$$



Figure 8: Three measures of a sample composition generated using the HBPL framework. The full composition and others can be found online at popstar.cs.byu.edu.

In essence we decompose a model of music *creation* to individually model the inspiration for the artefact, the symbolic (abstract) representation of the artefact, and the concrete rendering of the artefact.

Inspiration, $P(\iota)$ *Inspiration* (i.e., the method for deriving intention) may be more closely related to an artist’s or system’s “creative spark”. For example, observers often perceive greater creativity in artefacts which in some way relate to them or to their culture (Colton 2008). In the joint probability distribution on inspirations ι , compositions γ , and renderings ρ^m , we define $P(\iota)$ not as the distribution over intentions, but as the distribution over inspiring sources for the intention. In other words, not “what was the artefact intended to communicate?”, but “what was the inspiring *source* for what the artefact intended to communicate?”

In general this demonstrates an unanticipated benefit of factorization: we can condition on any variable that could be argued to influence the artefact’s creation. Many creative systems implicitly define inspiration based on the corpora that the data trains on. With the concept learning framework, we can model this attribute explicitly.

This represents an aspect not present in the model originally presented by Lake et al. (2015): not only are we modeling *what* artefacts can be generated, but also *why* they are generated. One possible way to model inspiration is to use an observer’s environment or culture as an inspiring source. Research in electroencephalogram-based affective computing (i.e., reading brain waves) suggests that computers may soon be able to perceive an observer’s emotional state beyond those of their human counterparts (Volioti et al. 2016). Alternatively, inspiration could be modeled using sentiment analysis in a variety of online domains. We plan to explore models of inspiration further in future research.

Rendering, $P(\rho^m|\iota, \gamma)$ The example model $P(\gamma|\iota)$ described above defines symbolic lyrical compositions (i.e., a leadsheet). However, evaluating an abstract artefact generally requires a concrete rendering of the artefact, whose distribution we model as $P(\rho^m|\iota, \gamma)$. As a proof of concept, we implemented and trained the described HBPL model on a small corpus of hand-annotated lyrical pop composition data. To concretely render compositions created using this model, we generated both printed sheet music (e.g., Figure 8) and an MP3 audio recording². Our MP3 audio file features computer-sung lyrics accompanied by synthesized

²audio recordings can be found at popstar.cs.byu.edu

piano and bass comping chords³.

Implications for Recommendation Systems Lake et al. present the model of $P(\psi)$ given in equation 1 as a submodel of the factoring of the joint probability distribution on character types ψ , tokens ω^m , and binary images I^m (2015):

$$P(\psi, \theta^1, \dots, \theta^M, I^1, \dots, I^M) = P(\psi) \prod_{m=1}^M P(I^m|\theta^m)P(\theta^m|\psi).$$

This means that given an image, the system can discover the motor program (i.e., abstract character type) that most likely generated it. This allows the system to one-shot classify and generate pairs of images that represent the same character type (specific examples of which were not seen in training).

By analogy, a model for $P(\gamma)$ (similar to $P(\gamma|\iota)$ just described) could be inserted into a joint probability on composition types γ , arrangements α^m , and audio recordings ρ^m ,

$$P(\gamma, \alpha^1, \dots, \alpha^M, \rho^1, \dots, \rho^M) = P(\gamma) \prod_{m=1}^M P(\rho^m|\alpha^m)P(\alpha^m|\gamma).$$

The implications of this model are more broadly significant: the HBPL framework is capable of inferring abstract representations of concrete artefacts, representations which more directly define meaning, composition, and causality. This is significant for two reasons. First, in some realms of creativity, simply deriving the abstract representation of an artefact is valuable (e.g., automatically transcribing sheet music from audio). Second, having an abstract representation allows concrete artefacts to be compared according to symbolic, conceptual criteria (e.g., recommendation systems based on meaning, or in the case of music, harmony, melodic pitch or rhythm, etc.). Though work has been done to approximate $P(\alpha^m|\gamma)$ (Benetos et al. 2013), effective comparison of artefacts hinges on the other terms in the factorization, $P(\gamma)$ and $P(\rho^m|\alpha^m)$, which are lacking.

Fitness and Self-Evaluation

The HBPL framework is designed to restrict the generation process *in situ* to produce only meaningful artefacts (as compared to a generate-and-test procedure). As discussed by Ventura (2016), this “baked-in” self-evaluation mechanism has the added benefit of being able to explain to some extent both the novelty, value, and motivation behind generated artefacts. Given its ability to compute probabilities, the HBPL framework could thus also be potentially leveraged as a fitness function for other types of generative models.

³generated using Harmony Assistant (v9.7.0f) and Virtual Singer (v3.2)

Big (Need for) Data

Any empirically-driven model requires training on a dataset representative of the artefact domain. Even if we had digital access to all of the compositions ever written, it would represent an infinitesimal portion of the songs that *could* be written. This is a challenge in many machine learning domains. Unique to the pop music domain, however, is that data is highly proprietary. What *is* available is extremely limited and of relatively poor quality. Compared to natural language, artefacts in music generally require relatively complex representations and relatively few possess the domain knowledge required to generate or transcribe the needed data. Among those who *do* understand and use it, music formatting can vary wildly and inexactly—creating additional challenges for a by-the-bit computer parser. Computers will only learn to speak music as quickly as we either formalize and ubiquitize the language of music *or* endow computers with AI tools to fill in the gaps on their own.

The particular challenge of accessing high-quality symbolic *pop* music datasets is significant. There is a dearth of well-annotated resources for those interested in studying any or all of the aspects of pop music composition. There is, however, much we can do to improve the situation. First, we need to make resources that *are* available more accessible (guitar tabs, lyrics sites, beatles). Second, we need to establish a better case for how society and industries stand to benefit from computational pop music research in order to generate a productive dialogue for the support and collaboration of those in possession of large pop music datasets (sheet music sites, spotify, etc., asking for APIs, etc). Note that this is different than asking them to simply give us their proprietary data. Third, we can do more to recognize contributions of novel datasets.

Conclusion

HBPL is a powerful framework for accomplishing tasks in computational creativity. Using principles of compositionality, causality, and learning-to-learn, such models are able to effectively learn and generate examples of complex creative concepts. Its probabilistic framework lends itself well to modeling important aspects of creativity such as inspiration and intention. The HBPL framework by nature compels researchers in domain-specific subareas of computational creativity to engage in the debates that the artists themselves are having, namely “how should an artefact be created?” and “does it matter?” To the extent that these challenges are effectively addressed on the scale of defining and training subconcept models, the HBPL model represents a useful framework for designing and assessing creative systems.

References

Barbieri, G.; Pachet, F.; Roy, P.; and Esposti, M. D. 2012. Markov constraints for generating lyrics with style. In *Proceedings of the Twentieth European Conference on Artificial Intelligence*, 115–120. IOS Press.

Bay, B.; Bodily, P.; and Ventura, D. 2017. Deterministic text transformation via constrained vector-word representa-

tion. To appear in Proceedings of the Eighth International Conference on Computational Creativity.

Benetos, E.; Dixon, S.; Giannoulis, D.; Kirchhoff, H.; and Klapuri, A. 2013. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems* 41(3):407–434.

Bodily, P., and Ventura, D. 2017. Musical metacreation using the hierarchical Bayesian program learning framework. Submitted to MUME2017.

Colton, S. 2008. Creativity versus the perception of creativity in computational systems. In *AAAI Spring Symposium: Creative Intelligent Systems*, volume 8.

Englemann, S., and Bruner, E. 1974. *DISTAR: Reading Level I*. Chicago: Science Research Associates.

Hirjee, H., and Brown, D. 2010. Using automated rhyme detection to characterize rhyming style in rap music. *Empirical Musicology Review* 5(4).

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Meyer, L. B. 2008. *Emotion and Meaning in Music*. University of Chicago Press.

Morris, R. G.; Burton, S. H.; Bodily, P. M.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *Proceedings of the Third International Conference on Computational Creativity*, 119–125.

Pachet, F., and Roy, P. 2014. Imitative leadsheet generation with user constraints. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, 1077–1078. IOS Press.

Pachet, F.; Roy, P.; Barbieri, G.; and Paris, S. C. 2011. Finite-length Markov processes with constraints. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 635–642.

Paris, C.; Swartout, W. R.; and Mann, W. C. 2013. *Natural language generation in artificial intelligence and computational linguistics*, volume 119. Springer Science & Business Media.

Steedman, M. J. 1984. A generative grammar for jazz chord sequences. *Music Perception: An Interdisciplinary Journal* 2(1):52–77.

Ventura, D. 2016. Mere generation: Essential barometer or dated concept? In *Proceedings of the Seventh International Conference on Computational Creativity*, 17–24.

Volioti, C.; Hadjidimitriou, S.; Manitsaris, S.; Hadjileontiadis, L.; Charisis, V.; and Manitsaris, A. 2016. On mapping emotional states and implicit gestures to sonification output from the ‘intangible musical instrument’. In *Proceedings of the Third International Symposium on Movement and Computing*, 30. ACM.

How Can We Deal With The Design Principle Of Visibility In Highly Encapsulated Computationally Creative Systems?

Liam Bray

Art and Design

The University of New South Wales
Cnr Oxford St and Greens Rd
Paddington NSW 2021 Australia
liam.bray@sydney.edu.au

Oliver Bown

Art and Design

The University of New South Wales
Cnr Oxford St and Greens Rd
Paddington NSW 2021 Australia
o.bown@unsw.edu.au

Benjamin Carey

Sydney Conservatorium of Music

The University of Sydney
1 Conservatorium Rd
Sydney NSW 2000 Australia
benjamin.carey@sydney.edu.au

Abstract

In this paper we analyse three specific computationally creative interface categories; direct manipulation systems, programmable interfaces and highly encapsulated systems. We conduct a preliminary investigation into a single expert user's experience of using tools which are designed for musical composition. Then discuss the implications encapsulation has on visibility in different computationally creative scenarios. Our analysis of the user's experience is then used to iteratively inform our categorisation of computationally creative interface types.

Introduction

Computationally creative (CC) music systems that are designed to be used as part of a co-creative process face a challenge in Norman's principle of Visibility (Norman 1988), a quality used in the evaluation of user experience. Our research suggests that as complexity increases, designers are forced to either allow their interfaces to become visually complex or heavily encapsulated (Bray and Bown 2016).

We begin by considering the context in which different users work. End-user programming has developed to become a feature of general-purpose computing (Blackwell 2002). It is commonplace to see non-professional programmers tasked with the specification, design, testing and maintenance of a spreadsheet of non-trivial complexity. We would argue that CC processes are also part of this development. CC software, once only accessible to domain specialists, is now being used to facilitate creative exploration in end-user oriented software. Our research aims to engage with this shift and facilitate a practice-centered approach to the evaluation of CC systems.

In this paper we provide a practice-centered evaluation of several CC music composition systems. The third author plays the role of an expert practitioner to research end-user oriented workflow when using these systems, and provide an analysis of the experience of working with these tools to achieve creative outcomes. We build on our previous work with a categorisation of CC interface types and discuss usability issues within CC systems with the goal of offering a heuristic solution to potential visibility issues. Our central research question is, what are the user's obstacles or frustrations when working with a highly encapsulated CC system?

Practice-based research can include the study of creative methods and tools from the perspective of the first person practitioner. One example of this is rather than conducting studies with users, the practice-based researcher becomes her own user and engages in a cycle of iterative research and practice studies (Smith and Dean 2009). This has the disadvantage of the potential failure to be objective, but the advantage of a more fluid, rapid and intuitive approach to the study of systems.

In previous work, we proposed that visibility in music systems, both digital and physical, can be successfully understood by thinking about the system in terms of a breakdown between the system's structure and some form of trajectory through that structure. This framework, based on interaction design principles, helps us better understand how users are capable of maintaining functional cognitive models of computationally complex systems and when that model breaks down in the context of musical interfaces. For example, dropping a pinball into a pinball machine, we think of the fixed structure of the pinball machine layout dictating the trajectory of the pinball. We think of all traditional acoustic instruments as having specific fixed structures around which a musician denotes a trajectory (Bray and Bown 2016).

In this paper we seek to expand this framework to include a user's capacity to conceptually map abstraction. Here we draw on Blackwell (Blackwell 2002), who in turn draws on Lindsay (Lindsay 1988), to provide the conceptual foundation required for us to create a practical framework for the design of musical interfaces:

If a planning agent maintains a mental representation of the situation in which it acts, the process of planning relies on the agent being able to simulate updates to the situation model, in order to evaluate the results of potential actions. (Blackwell 2002)

CC systems are often heavily encapsulated systems which, for practical reasons, intentionally hide their complexity. This in turn can prevent the user from developing a coherent mental representation of the system's intended state. By way of example, consider the use of an algorithm such as an artificial neural network (ANN) in a CC system. With 2 inputs, 4 hidden and 2 output nodes, the inherent complexity of this algorithm may cause the designer to question the utility of representing the algorithm's internal state

to the user. In such a scenario, the decision to represent the ANN may be based upon the designer's belief that a user is able to maintain a mental model of the effect a modification to an input node will have on the system when it updates. This example implies that it is necessary for the user to be aware of what the ANN is doing as it completes a task.

On the other hand, abstraction and encapsulation are actually common in design. For example, if we conduct an image search which relies on a ANN, is it necessary for the user to have mental representation of the process by which the ANN has completed the search? Arguably (Dennett 1989), abstraction does not inherently inhibit usability in CC systems. Our observation is that in compositional tasks in which a user is engaged with a CC system, intelligible actions are facilitated by structural knowledge of the system's process.

This type of abstraction, described by Blackwell (Blackwell 2002) as the loss of direct manipulations', is the same problem that programmable notation systems are challenged by. Blackwell has described fundamental conceptual limitations of non-direct manipulations in general purpose programming as abstraction over time and abstraction over a class of situations (Blackwell and Green 2003).

Computationally Creative Interfaces

The pay-off for encapsulation in general-purpose programming is based on the paradigm of productivity through automation (Wilkes 1956; Gaver 2002). Blackwell (Blackwell 2002) demonstrates this with the establishment of a cost-benefit analysis. Cost being the amount of effort cognitive or otherwise weighed against the benefit of having some given thing automated.

Programmable computationally creative interfaces, understood in the context of creativity support tools (CSTs) (Shneiderman 2007) are challenging when placed in this paradigm. Blackwell proposes that for tasks that have the potential to be automated, users have to weigh up the cognitive effort and risk associated with setting up an automated approach. He calls this "prospecting", as in "digging a hole in the ground to find out whether it is worth siting a gold mine there". Direct manipulation, in this scenario, can work against the future benefit of having a process automated. For example, when using a non-realtime co-creative music system, the user takes on the role of curator or editor, often leading the user towards a more structural model of control.

To further describe CC specific interfaces we will use three high level categories of CC interface types.

- Direct manipulation systems: Provides the user with an immediate representation of objects of interest that the user can manipulate with immediate effect.
- Programmable interfaces: A notational interface which allows the user to define a set of control commands to be executed as a program. Objects may be represented directly or encapsulated in this process.
- Highly encapsulated systems: Systems in which the representation of the process is wholly encapsulated. Users are presented with parameterised abstract control of the system, which has hidden underlying processes.

We suggest however that this spectrum is context dependent as many interfaces have multiple representative or notational models which can be adapted in specialist scenarios. This is a common feature described as "abstraction gradient" in the cognitive dimensions of notations usability evaluation (Green and Petre 1996) which provides design principles for notations, user interfaces and programming languages. Here abstraction gradient can be understood as the fluid description of the state the system is currently in.

These high level categories of CC interface types provide a context for our evaluation. We will draw on this categorisation to develop an analysis of the user's experience when engaged in using a CC system in an open ended creative task.

Methodology

We have conducted a preliminary investigation into a single user's experience of using tools which are designed for musical composition that demonstrate non-direct manipulations in compositional workflow. To define our methodological approach to this investigation we draw on Candy's (2006) guide to practice-based research. Our goal is to better understand artists practice when engaged in compositional tasks with CC systems. In this way we are primarily concerned with contributing to operationally significant knowledge within the practice of algorithmic music composition.

In approaching this goal we reflect on previous studies conducted by the authors (Bray and Bown 2014). Previously we implemented a study based on the Cognitive Dimensions of Notations framework (Blackwell and Green 2003) which focuses on comparing the users' reflections on the usability of computer based systems by identifying conceptual models employed by notational systems. Here we draw on a similar process in order to gain insight into the user's experience of using a group of systems which exhibit specific properties.

In this paper, the 3rd author, Benjamin Carey, was enlisted specifically to play the role of an expert user. The 1st and 3rd author are both engaged in practice with CC systems. Evaluating usability is an extension of both 1st and 3rd authors practice-based research on these systems. Carey was not involved in the preparation of the paper or the development of the contexts discussed here but did assist in editing and has been included as an author to represent a collaborative research approach rather than a blind user-study style approach.

As Carey's contribution to this analysis is the primary source of knowledge, we consider his role as the user to be that of a practice-based researcher, reflecting on his experience using three Max for Live based composition tools in Ableton Live. Our analysis draws on his results to further iterate and inform the parameters of our interface categorisation. Our analysis below outlines our reflections on his responses.

Each of these tools were selected as they represent a different functional workflow, a different level of abstraction and a varying scale of complexity in algorithmic processing, but by definition demonstrate non-direct manipulations in compositional workflow. Carey answered a single ques-

	Direct Manipulation System	Programmable Interfaces	Highly Encapsulated System
System's use case	Skeuomorphic or systematic composition or hierarchies.	End-user specification of CC processes.	Generation of unique or exploratory content. Autonomous creation.
System's algorithm type	Networks, topologies, logic maps.	Typographic or object oriented syntaxes.	AI based processes, eg. neural networks and machine learning.
System's algorithmic complexity	Requires sufficient notational input to become complex.	Scaleable, can become very complex through encapsulation but typically objects will remain accessible.	Simple algorithms can be highly encapsulated. But end user oriented system's, that use AI based processes are often in this category.
System aids creative exploration	Exploratory results will rely on object interactions being sufficiently complex.	User can create highly specialised processes. Enabling specification of desired search space.	Will typically create a vast amount of output with minimal or no input.
System's ability to (typically) generate immediate results	System output is immediate and tangible.	System requires design or specification prior to event generation.	System output is immediate and intangible.
User has the ability to predict outcomes	Until the system is sufficiently complex the user will be able to visually discern the consequence of actions.	Conceptual capacity to discern actions in encapsulated interactions will decrease with system complexity.	The system provides minimal tangible means to discern predictable outcomes. But the user may still have a good mental model.
User is able to rapidly visualise outcomes	High visible system interactions.	Structural control will enable rapid visual outcomes, but encapsulated interactions will be hidden as the design becomes more abstract.	Minimal visualisation of processes is provided, speed of system output could vary, but some system's will provide real-time outcomes.
User is able to directly manipulate or program the outcome.	Yes, highly literal interface.	Initially, but with encapsulation this will shift toward structural control.	Structural control of process only.
User is able to iterate between the use of the algorithm and direct manipulation	Direct Manipulation only.	Yes, the interface is mutable.	Algorithmic control only, often preset or range based input variables.

Figure 1: Categorisation of CC interfaces

tionnaire for each system he used. Below we outline each system, and discuss his responses.

Description of Systems

Jnana Live is an algorithmic musical accompaniment tool, it allows the user to analyse realtime MIDI information coming from Ableton Live. Based on this input, through an algorithmic process it creates a model to generate unique material in a similar style. Jnana has a second system called Jnana Clips which is used to analyse existing MIDI information in the form of Ableton Live 'Clips'. In this study we only looked at the 'Live' device which can functionally operate in a similar manner when MIDI information is routed as a real-time input to the device.

Controls that are represented to the user are separated into three groups, Input Analysis expressed as 'when' and 'how'. 'Response generation' allowing for the modification of manual or automatic response generation and lastly 'Other' providing control of MIDI passthrough. The primary controls of the tool are found in the input analysis section. Under which we find hold/auto to analyse, initiating the model. Phrase detection, where a time in (ms) can be

specified which determines how much silence is required for an input 'phrase' to have occurred. Under how, 'use starting statistics' allows the user to control if the start of each phrase in the analysis will be considered when generating new phrases. Lastly 'assume circular' which loops the analysis of phrases, optimising it for more consistent loop-based inputs.

Patter is a stochastic event generator, it generates 'segments' of MIDI events. A segment has a set duration and the events within that segment are stochastically generated based on the weightings specified in the rhythm, accent and pitch sections of the interface. Patter will by default begin generating events based on the global transport settings in Ableton Live and output them to a MIDI channel, assignable in Ableton Live to any desired MIDI input source such as a virtual instrument. Patter also includes the functionality to be slaved to other instances of Patter enabling it to play with or after segments generated by those other instances. Segments can be looped based on a weighted probability and are segments are divided into six sections as represented in the loop section.

Within the Rhythm section of the interface are four gener-

How would you describe your approach to working with the system?	Does the system fit into your workflow in Live?	Do you trust the tool to make good musical suggestions?	What functionality did the system provide?
Was that functionality a good thing?	Did you feel that you had control over the tool?	Detail in your own words any interesting or surprising interactions you had with the system.	Did the musical content proposed by the system ever change the musical direction you were heading in?
How often do you think the system proposed interesting results?	Did you feel that you were passive in the interaction with the system?	Did your interactions with the system lessen your desire to maintain control of musical direction?	Did the system show signs of musical structure?

Figure 2: Excerpt from Questionnaire

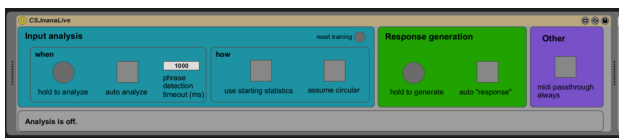


Figure 3: Jnana Live

ators with different ranges, each distribution includes mean and deviation which allows the user to shape the events generated in each segment. The First takes the tempo from Ableton Live's transport and provides a distribution from 8th notes to Whole notes. The second generator multiplies or divides that beat, based on the selection of the \times or \div operator, providing the user with the ability to achieve more complex rhythmic events. For example, multiplying by 3 yields "dotted" notes, dividing by 3 yields 'triplets'. The next generator determines the number of notes and the next determines the number of rests in a segment. Additionally the user is able to select 'post' so that rests will occur after the notes, not before. Also an option for 'downbeat' is provided meaning segments will always begin on a downbeat according to their rhythm. The Accent generator determines where the 'accent' (highest velocity) is placed within the segment. The Pitch region allows the user to define available notes, and toggle between midi input or a generator weighted from high to low.

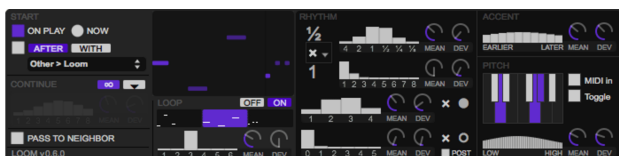


Figure 4: Patter

Style Machine Lite, produces complete musical pieces that are modelled on a corpus and a groove. It uses Ableton Lives clip view to populate short sections of a song across

a range of channels defined by their intended instrument or sound type. Style Machine Lite produces long term musical structures by populating clips in sequence. Users are able to access Ableton Live's complete functionality when the clips have been populated, allowing for editing of sounds and clips. The user is provided with pre-set styles which are focused on predominant electronic music genres. And grooves which are musically descriptive. The user can control three parameters complexity, density and length. These controls function to shape the structure of the generated material and are applied across either a generated phrase that is, a single row of clips, or the tracks entirety.



Figure 5: Style Machine Lite

Analysis

After a self-determined amount of time using each tool, Carey provided answers to the questionnaire we provided him. Firstly, we were interested in understanding if he was able to demonstrate a tangible understanding of the tool he was using. In all three systems, Carey was able to identify the primary functionality and navigate the interface proficiently enough to generate output from it. He described his approach to working with the system in each instance. In both Jnana and Style machine, Carey was focused on manipulating parameters, so that the system would provide feedback for him to evaluate before making additional adjustments. In this way he was attempting to build a functional model of the effect of a given parameter manipulation. One consideration here is that this process is cognitively demanding as it relies on the user to monitor the interface and the

	Patter	Jnana Live	Style Machine Lite
System's use case	Rhythm and melody generator.	Musical accompaniment or variation tool.	Corpus based song generation.
System's algorithm type	Stochastic.	Simple hidden Markov models.	Complex Markov processes.
System's algorithmic complexity	Simple statistical distributions.	Complex real-time analysis.	Complex corpus analysis.
System aids creative exploration	Weighted random events. Might allow specification of exploratory outputs.	Will iterate on input data in real time. Providing divergent variation on rhythm and melody.	Creates complete complex musical structures.
System's ability to (typically) generate immediate results	System will output when transport is started.	System will begin generating output when input is present and analysis begun.	System creates output when generation is started.
User has the ability to predict outcomes	User expressed ability to discernibly control system output in a nuanced way.	User described constraining the system to generate variations.	User relied on notational descriptions to navigate proposed results.
User is able to rapidly visualise outcomes	User was able to act on visual feedback provided by the system's interface.	User described a sense of detachment between the choices made and outputs generated. But was positive in regards to it's results.	User was not able to visualise outcomes as they were generated.
User is able to directly manipulate or program the outcome.	Yes, user described the interface as mutable and tangible.	No, the user perceived structural parameterised control only.	No, the user perceived structural parameterised control only.
User is able to iterate between the use of the algorithm and direct manipulation	Somewhat. User acted to refine parameters to create desired algorithmic output.	No, user generated output based on parameterised control.	No, user generated output based on parameterised control.

Closest Categorisation Correlation	Direct manipulation	Programmable interface	Highly encapsulated system

Figure 6: Analysis of user responses based on CC Interface Categorisation

systems output precisely. Another is that this initial processes stimulates undirected search as the parameter space is unknown to the user.

In Patter, Carey cited the simplicity of the generation in the system as exploratory, and began searching for usable compositional content. This is distinct from Jnana and Style machine, Carey's sense of immediacy in controlling the system enabled him to begin a process of directed search. In this way we could consider Patter an example of a highly literal interface. From his answers we can anecdotally establish the nature of the Carey's capacity to model each system, or at least establish a description of the strategy they formed when using the system. In this regard, he was successful in determining the intended design of each tool. When asked a supplementary question about Jnana's functionality, Carey articulates the precise paradigm CC systems are faced with.

"The limited information and parameters to tweak makes me more likely to use something like this. Endless parameters in something as complex as a Markov model is daunting, even for someone like myself with experience with such approaches."

This is a highly opaque system, which conventional user experience knowledge tells us can be situationally problematic. Norman's principle of visibility being the primary articulation of this (Rogers, Preece, and Sharp 2007; Bray and Bown 2016; Norman 2013). We infer that Carey requires non-direct, abstract control of the system to be able to make intelligible actions within it. However this has a clear trade-off. In both Jnana and Style Machine Lite, Carey expressed that he did not feel as if he had strong control over the system's output.

”I felt passive as it’s difficult to see a real connection between these choices of material and the output. Though this may change over time.”

Highly encapsulated systems can also represent a conceptual black box (Kolen and Pollack 1994) to the user. Jnana Live’s highly encapsulated interface, focused on initiation and structural control of the algorithm is suggestive of its intended purpose as an accompaniment system. The absence of parameterised control of temporal event or pitch information also embodies a highly exploratory approach to compositional workflow which could correlate with existing research on the open-ended nature of creative discovery (Saunders and Gero 2002). The system allowed the user to create endless variations, however this was at the cost of a sense of passivity in the compositional process.

Feeling passive may not be desirable from a user’s perspective, but it does not necessitate that in a co-creative situation where both agents are enabled to act on each other that you cannot be musically successful. Carey describes being able to successfully input midi information into the system and generate outcomes which he considered compositionally desirable. But by contrast Style Machine Lite, which is also highly encapsulated took what Carey perceived as an undesirable amount of control away from him. He described the system’s output as impressive, but was unable to discernibly act on the system to navigate towards desired compositional output. We could possibly describe this characteristic as providing a sense of system autonomy or encouraging passivity. This

Out of all three systems, only Patter provides the user with a visual interface that depicts musical events as objects. But as the output of these systems is auditory, Carey was still able to develop a cognitive understanding of the effect notational changes made to the systems output. Or at least was forced to rely on this feedback in an attempt to build a cognitive understanding. In this sense, the system’s output is not dissimilar to an auditory display (Walker and Kramer 1996). One possible benefit of this could be that auditory feedback afforded Carey and additional means by which to determine the system’s trajectory, enabling him to better understand the system’s structure and act to steer the algorithm in a desired direction.

We can also consider these interfaces in the context of a syntactic/semantic model of user behaviour (Shneiderman 1983). That is, that there are two kinds of knowledge represented by software systems:

First, the user must possess syntactic knowledge, which correlates to valid input methods or permissible delimiters. In an direct manipulation system interface this could be valid topographic arrangements or object manipulations. In a programmable interface this is easily identified as valid syntactic statements. In a highly encapsulated system this could just be successfully inputting information. This type of knowledge, may not be entirely system dependant but will feature idiosyncratic notations making it harder to recall for new or infrequent users.

And second semantic knowledge, which is acquired

through analogy, example or generalised conceptual principles. Shneiderman places this type of knowledge on a scale from low level program domain actions to high level problem domain features. Here an direct manipulation interface might provide skeuomorphic analogy, a programmable interface interface might facilitate a process-based hierarchy of events and a highly encapsulated system might draw on cultural knowledge through simple iconography.

Conclusion

In conclusion, we have discussed issues surrounding the use of highly encapsulated CC systems. Employing the use of a high level categorisation of interface types allowing us to discern more clearly what Carey our practice-based researcher, was experiencing when using each system. A users capacity to understand and cognitively map the structure or trajectory of an interface relies on the user being able to gain access to intelligible forms of feedback. The user’s perceived experience of the opaqueness of the system’s feedback can be analysed through practice lead research as we have demonstrated. This research intends to contribute to and inform the design of CC systems to make them more usable and in turn more useful for creative practitioners.

References

- Blackwell, A., and Green, T. 2003. Notational systems—the cognitive dimensions of notations framework. *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science*. Morgan Kaufmann.
- Blackwell, A. 2002. What is programming. In *14th workshop of the Psychology of Programming Interest Group*, 204–218.
- Bray, L., and Bown, O. 2014. Linear and non-linear composition systems: User experience in nodal and pro tools. In *Proceedings of the Australian Computer Music Association Conference*.
- Bray, L., and Bown, O. 2016. Applying core interaction design principles to computational creativity. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Dennett, D. C. 1989. *The intentional stance*. MIT press.
- Gaver, B. 2002. Designing for Homo Ludens, Still. *Interaction Research Studio, Goldsmiths, University of London, I3 Magazine No. 12* 163–178.
- Green, T. R. G., and Petre, M. 1996. Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework. *Journal of Visual Languages & Computing* 7(2):131–174.
- Kolen, J. F., and Pollack, J. B. 1994. The observers’ paradox: Apparent computational complexity in physical systems. *The Journal of Experimental and Theoretical Artificial Intelligence*.
- Lindsay, R. K. 1988. Images and inference. *Cognition* 29(3):229–250.
- Norman, D. 1988. *The Design of Everyday Things*. New York: Basic Books.

- Norman, D. A. 2013. *The design of everyday things: Revised and expanded edition*. Basic books.
- Rogers, Y.; Preece, J.; and Sharp, H. 2007. Interaction design.
- Saunders, R., and Gero, J. S. 2002. How to study artificial creativity. In *Proceedings of the 4th conference on Creativity & cognition*, 80–87. ACM.
- Shneiderman, B. 1983. Human factors of interactive software. In *IBM Germany Scientific Symposium Series*, 9–29. Springer.
- Shneiderman, B. 2007. Creativity Support Tools: Accelerating Discovery and Innovation. *Communications of the ACM* 50(12).
- Smith, H., and Dean, R. 2009. *Practice-led research, research-led practice in the creative arts*. Edinburgh University Press.
- Walker, B. N., and Kramer, G. 1996. Mappings and metaphors in auditory displays: An experimental assessment. Georgia Institute of Technology.
- Wilkes, M. V. 1956. Automatic digital computers..

A Unit Selection Methodology for Music Generation Using Deep Neural Networks

Mason Bretan

Georgia Institute of Technology
Atlanta, GA

Gil Weinberg

Georgia Institute of Technology
Atlanta, GA

Larry Heck

Google Research
Mountain View, CA

Abstract

Several methods exist for a computer to generate music based on data including Markov chains, recurrent neural networks, recombination, and grammars. We explore the use of unit selection and concatenation as a means of generating music using a procedure based on ranking, where, we consider a unit to be a variable length number of measures of music. We first examine whether a unit selection method, that is restricted to a finite size unit library, can be sufficient for encompassing a wide spectrum of music. This is done by developing a deep autoencoder that encodes a musical input and reconstructs the input by selecting from the library. We then describe a generative model that combines a deep structured semantic model (DSSM) with an LSTM to predict the next unit, where units consist of four, two, and one measures of music. We evaluate the generative model using objective metrics including mean rank and accuracy and with a subjective listening test in which expert musicians are asked to complete a forced-choice ranking task. Our system is compared to a note-level generative baseline model that consists of a stacked LSTM trained to predict forward by one note.

Introduction

For the last half century researchers and artists have developed many types of algorithmic composition systems. These individuals are driven by the allure of both simulating human aesthetic creativity through computation and tapping into the artistic potential deep-seated in the inhuman characteristics of computers. Some systems may employ rule-based, sampling, or morphing methodologies to create music (Papadopoulos and Wiggins 1999). We present a method that falls into the class of symbolic generative music systems consisting of data driven models which utilize statistical machine learning.

Within this class of music systems, the most prevalent method is to create a model that learns likely transitions between notes using sequential modeling techniques such as Markov chains or recurrent neural networks (Pachet and Roy 2011; Franklin 2006). The learning minimizes note-level perplexity and during generation the models may stochastically or deterministically select the next best note given the preceding note(s).

All rights reserved.

In this paper we describe a method to generate monophonic melodic lines based on unit selection. The approach is inspired by 1) the theory that jazz improvisation predominantly consists of inserting and concatenating predetermined musical structures or note sequences (Norgaard 2014; Pressing 1988) and 2) techniques that are commonly used in text-to-speech (TTS) systems. The two system design trends found in TTS are statistical parametric and unit selection (Zen, Tokuda, and Black 2009). In the former, speech is completely reconstructed given a set of parameters. The premise for the latter is that new, intelligible, and natural sounding speech can be synthesized by concatenating smaller audio units that were derived from a preexisting speech signal (Hunt and Black 1996; Black and Taylor 1997; Conkie et al. 2000). Unlike a parametric system, which reconstructs the signal from the bottom up, the information within a unit is preserved and is directly applied for signal construction. When this idea is applied to music, the generative system can similarly get some of the structure inherent to music “for free” by pulling from a unit library.

The ability to directly use the music that was previously composed or performed by a human can be a significant advantage when trying to imitate a style or pass a musical Turing test. However, there are also drawbacks to unit selection that the more common note-to-note level generation methods do not need to address. The most obvious drawback is that the output of a unit selection method is restricted to what is available in the unit library. Note-level generation provides maximum flexibility in what can be produced. Ideally, the units in a unit selection method should be small enough such that it is possible to produce a wide spectrum of music, while, remaining large enough to take advantage of the built-in information.

Another challenge with unit selection is that the concatenation process may lead to “jumps” or “shifts” in the musical content or style that may sound unnatural and jarring to a listener. Even if the selection process accounts for this, the size of the library must be sufficiently large in order to address many scenarios. Thus, the process of selecting units can equate to a massive number of comparisons among units when the library is very big. Even after pruning this can be a lot of computation. However, this is less of an issue as long as the computing power is available and unit evaluation can be performed in parallel processes.

In this work we explore unit selection as a means of music generation. We first build a deep autoencoder where reconstruction is performed using unit selection. This allows us to make an initial qualitative assessment of the ability of a finite-sized library to reconstruct never before seen music. We then describe a generative method that selects and concatenates units to create new music.

The proposed generation system ranks individual units based on two values: 1) a semantic relevance score between two units and 2) a concatenation cost that describes the distortion at the seams where units connect. The semantic relevance score is determined by using a deep structured semantic model (DSSM) to compute the distance between two units in a compressed embedding space (Huang et al. 2013). The concatenation cost is derived by first learning the likelihood of a sequence of musical events (such as individual notes) with an LSTM and then using this LSTM to evaluate the likelihood of two consecutive units. We evaluate the model’s ability to select the next best unit based on ranking accuracy and mean rank. We use a subjective listening test to evaluate the “naturalness” and “likeability” of the musical output produced by versions of the system using units of lengths four, two, and one measures. We additionally compare our unit selection based systems to the more common note-level generative models using an LSTM trained to predict forward by one note.

Related Work

Many methods for generating music have been proposed. The data-driven statistical methods typically employ n-gram or Markov models (Chordia, Sastry, and Şentürk 2011; Pachet and Roy 2011; Wang and Dubnov 2014; Simon, Morris, and Basu 2008; Collins et al. 2016). In these Markov-based approaches note-to-note transitions are modeled (typically bi-gram or tri-gram note models). However, by focusing only on such local temporal dependencies these models fail to take into account the higher level structure and semantics important to music.

Like the Markov approaches, RNN methods that are trained on note-to-note transitions fail to capture higher level semantics and long term dependencies (Coca, Romero, and Zhao 2011; Boulanger-Lewandowski, Bengio, and Vincent 2012; Goel, Vohra, and Sahoo 2014). However, using an LSTM, Eck demonstrated that some higher level temporal structure can be learned (Eck and Schmidhuber 2002). The overall harmonic form of the blues was learned by training the network with various improvisations over the standard blues progression.

We believe these previous efforts have not been successful at creating rich and aesthetically pleasing large scale musical structures that demonstrate an ability to communicate complex musical ideas beyond the note-to-note level. A melody (precomposed or improvised) relies on a hierarchical structure and the higher-levels in this hierarchy are arguably the most important part of generating a melody. Much like in story telling it is the broad ideas that are of the most interest and not necessarily the individual words.

Rule-based grammar methods have been developed to address such hierarchical structure. Though many of these sys-

tems’ rules are derived using a well-thought out and careful consideration to music theory and perception (Lerdahl 1992), some of them do employ machine learning methods to create the rules. This includes stochastic grammars and constraint based reasoning methods (McCormack 1996). However, grammar based systems are used predominantly from an analysis perspective and do not typically generalize beyond specific scenarios (Lerdahl and Jackendoff 1987; Papadopoulos and Wiggins 1999).

The most closely related work to our proposed unit selection method is David Cope’s *Experiments in Musical Intelligence*, in which “recombinancy” is used (Cope 1999). Cope’s process of recombinancy first breaks down a musical piece into small segments, labels these segments based on various characteristics, and reorders or “recombines” them based on a set of musical rules to create a new piece. Though there is no machine learning involved, the underlying process of stitching together preexisting segments is similar to our method. However, we attempt to learn how to connect units based on sequential modeling with an LSTM. Furthermore, our unit labeling is derived from a semantic embedding using a technique developed for ranking tasks in natural language processing (NLP).

Our goal in this research is to examine the potential for unit selection as a means of music generation. Ideally, the method should capture some of the structural hierarchy inherent to music like the grammar based strategies, but be flexible enough so that they generalize as well as the generative note-level models. Challenges include finding a unit length capable of this and developing a selection method that results in both likeable and natural sounding music.

Reconstruction Using Unit Selection

As a first step towards evaluating the potential for unit selection, we examine how well a melody or a more complex jazz solo can be reconstructed using only the units available in a library. Two things are needed to accomplish this: 1) data to build a unit library and 2) a method for analyzing a melody and identifying the best units to reconstruct it.

Our dataset consists of 4,235 lead sheets from the Wikifonia database containing melodies from genres including (but not limited to) jazz, folk, pop, and classical (Simon, Morris, and Basu 2008). In addition, we collected 120 publicly available jazz solo transcriptions from various websites.

Design of a Musical Autoencoder

In order to analyze and reconstruct a melody we trained a deep autoencoder to encode and decode a single measure of music. This means that our unit (in this scenario) is one measure of music. From the dataset there are roughly 170,000 unique measures. Of these, there are roughly 20,000 unique rhythms seen in the measures. We augment the dataset by manipulating pitches through linear shifts (transpositions) and alterations of the intervals between notes resulting in roughly 80 million unique measures.

The intervals are altered using two methods: 1) adding a constant value to the original intervals and 2) multiplying a constant value to the intervals. Many different constant

values are used and the resulting pitches from the new interval values are superimposed on to the measure’s original rhythms. The new unit is added to the dataset. We restrict the library to measures with pitches that fall into a five octave range (midi notes 36-92). Each measure is transposed up and down a half step so that all instances within the pitch range are covered. The only manipulation performed on the duration values of notes within a measure is the temporal compression of two consecutive measures into a single measure. This “double time” representation effectively increases the number of measures, while leaving the inherent rhythmic structure intact. After all of this manipulation and augmentation there are roughly 80 million unique measures. We use 60% for training and 40% for testing our autoencoder.

The first step in the process is feature extraction and creating a vector representation of the unit. Unit selection allows for a lossy representation of the events within a measure. As long as it is possible to rank the units it is not necessary to be able to recreate the exact sequence of notes with the autoencoder. Therefore, we can represent each measure using a bag-of-words (BOW) like feature vector. Our features include:

1. counts of note tuples $\langle \text{pitch}_1, \text{duration}_1 \rangle$
2. counts of pitches $\langle \text{pitch}_1 \rangle$
3. counts of durations $\langle \text{duration}_1 \rangle$
4. counts of pitch class $\langle \text{class}_1 \rangle$
5. counts of class and rhythm tuples $\langle \text{class}_1, \text{duration}_1 \rangle$
6. counts of pitch bigrams $\langle \text{pitch}_1, \text{pitch}_2 \rangle$
7. counts of duration bigrams $\langle \text{duration}_1, \text{duration}_2 \rangle$
8. counts of pitch class bigrams $\langle \text{class}_1, \text{class}_1 \rangle$
9. first note is tied previous measure (1 or 0)
10. last note is tied to next measure (1 or 0)

The pitches are represented using midi pitch values. The pitch class of a note is the note’s pitch reduced down to a single octave (12 possible values). We also represent rests using a pitch value equal to negative one. Therefore, no feature vector will consist of only zeros. Instead, if the measure is empty the feature vector will have a value of one at the position representing a whole rest. Because we used data that came from symbolic notation (not performance) the durations can be represented using their rational form (numerator, denominator) where a quarter note would be ‘1/4.’ Finally, we also include beginning and end symbols to indicate whether the note is a first or last note in a measure.

The architecture of the autoencoder is depicted in Figure 1. The objective of the decoder is to reconstruct the feature vector and not the actual sequence of notes as depicted in the initial unit of music. Therefore, the entire process involves two types of reconstruction:

1. **feature vector reconstruction** - the reconstruction performed and learned by the *decoder*.
2. **music reconstruction** - the process of selecting a unit that best represents the initial input musical unit.

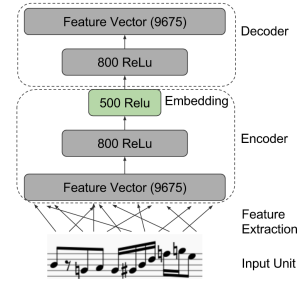


Figure 1: Autoencoder architecture – The unit is vectorized using a BOW like feature extraction and the autoencoder learns to reconstruct this feature vector.

In order for the network to learn the parameters necessary for effective feature vector reconstruction by the decoder, the network uses leaky rectified linear units ($\alpha = .001$) on each layer and during training minimizes a loss function based on the cosine similarity function

$$\text{sim}(\vec{X}, \vec{Y}) = \frac{\vec{X}^T \cdot \vec{Y}}{|\vec{X}| |\vec{Y}|} \quad (1)$$

where \vec{X} and \vec{Y} are two equal length vectors. This function serves as the basis for computing the distance between the input vector to the encoder and output vector of the decoder. Negative examples are included through a softmax function

$$P(\vec{R}|\vec{Q}) = \frac{\exp(\text{sim}(\vec{Q}, \vec{R}))}{\sum_{\vec{d} \in D} \exp(\text{sim}(\vec{Q}, \vec{d}))} \quad (2)$$

where \vec{Q} is the feature vector derived from the input musical unit, Q , and \vec{R} represents the reconstructed feature vector of Q . D is the set of five reconstructed feature vectors that includes \vec{R} and four candidate reconstructed feature vectors derived from four randomly selected units in the training set. The network then minimizes the following differentiable loss function using gradient descent

$$-\log \prod_{(Q,R)} P(\vec{R}|\vec{Q}) \quad (3)$$

A learning rate of 0.005 was used and a dropout of 0.5 was applied to each hidden layer, but not applied to the feature vector. The network was developed using Google’s *Tensorflow* framework (Abadi et al. 2016).

Music Reconstruction through Selection

The feature vector used as the input to the autoencoder is a BOW-like representation of the musical unit. This is not a loss-less representation and there is no effective means of converting this representation back into its original symbolic musical form. However, the nature of a unit selection method is such that it is not necessary to reconstruct the original sequence of notes. Instead, a candidate is selected from the library that best depicts the content of the original unit based on some distance metric.

Table 1: Results

mean rank @ 50	1.003
accuracy @ 50	99.98
collision rate per 100k	91

In TTS, this distance metric is referred to as the *target cost* and describes the distance between a unit in the database and the target it’s supposed to represent (Zen, Tokuda, and Black 2009). In our musical scenario, the targets are individual measures of music and the distance (or cost) is measured within the embedding space learned by the autoencoder. The unit whose embedding vector shares the highest cosine similarity with the query embedding is chosen as the top candidate to represent a query or target unit. We apply the function

$$\hat{y} = \arg \max_y \text{sim}(x, y) \quad (4)$$

where x is the embedding of the input unit and y is the embedding of a unit chosen from the library.

The encoding and selection can be objectively and qualitatively evaluated. For the purposes of this particular musical autoencoder, an effective embedding is one that captures perceptually significant semantic properties and is capable of distinguishing the original unit in the library (low collision rate) despite the reduced dimensionality. In order to assess the second part we can complete a ranking (or sorting) task in which the selection rank (using equation 5) of the truth out of 49 randomly selected units (rank@50) is calculated for each unit in the test set. The collision rate can also be computed by counting the instances in which a particular embedding represents more than one unit. The results are reported in the table below.

Given the good performance we can make a strong assumption that if an identical unit to the one being encoded exists in the library then the reconstruction process will correctly select it as having the highest similarity. In practice, however, it is probable that such a unit will not exist in the library. The number of ways in which a measure can be filled with notes is insurmountably huge and the millions of measures in the current unit library represent only a tiny fraction of all possibilities. Therefore, in the instances in which an identical unit is unavailable an alternative, though perceptually similar, selection must be chosen.

Autoencoders and embeddings developed for image processing tasks are often qualitatively evaluated by examining the similarity between original and reconstructed images (van den Oord et al. 2016). Likewise, we can assess the selection process by reconstructing never before seen music.

Figure 2 shows the reconstruction of an improvisation (see the related video for audio examples ¹). Through these types of reconstructions we are able to see and hear that the unit selection performs well. Also, note that this method of reconstruction utilizes only a target cost and does not include a concatenation cost between measures.

Another method of qualitative evaluation is to reconstruct from embeddings derived from linear interpolations between

¹<https://youtu.be/Bbyvb02F7ug>



Figure 2: The music on the staff labeled “reconstruction” (below the line) is the reconstruction (using the encoding and unit selection process) of the music on the staff labeled “original” (above the line).

two input seeds. The premise is that the reconstruction from the vector representing the weighted sum of the two seed embeddings should result in samples that contain characteristics of both seed units. Figure 3 shows results of reconstruction from three different pairs of units.

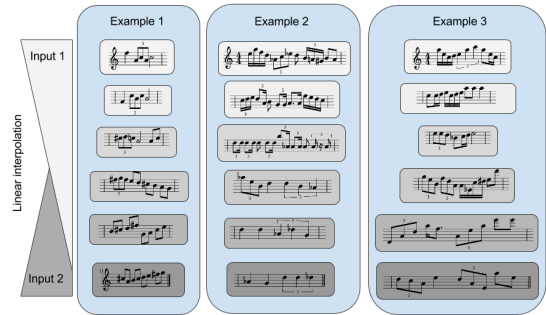


Figure 3: Linear interpolation in the embedding space in which the top and bottom units are used as endpoints in the interpolation. Units are selected based on their cosine similarity to the interpolated embedding vector.

Generation using Unit Selection

In the previous section we demonstrated how unit selection and an autoencoder can be used to transform an existing piece of music through reconstruction and merging processes. The embeddings learned by the autoencoder provide features that are used to select the unit in the library that best represents a given query unit. In this section we explore how unit selection can be used to generate sequences of music using a predictive method. The task of the system is to generate sequences by identifying good candidates in the library to contiguously follow a given unit or sequence of units.

The process for identifying good candidates is based on the assumption that two contiguous units, (u_{n-1}, u_n) , should share characteristics in a higher level musical semantic space (semantic relevance) and the transition between the last and first notes of the first and second units respectively

should be likely to occur according to a model (concatenation). This general idea is visually portrayed in Figure 4. We use a DSSM based on BOW-like features to model the semantic relevance between two contiguous units and a note-level LSTM to learn likely note sequences (where a note contains pitch and rhythm information).

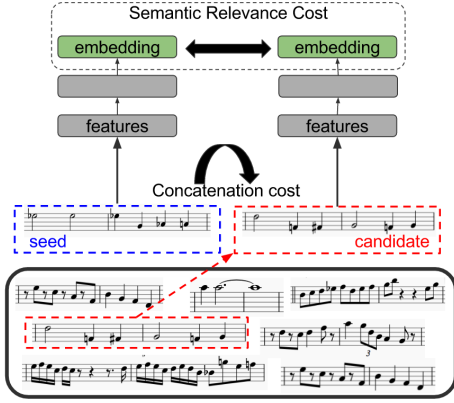


Figure 4: A candidate is picked from the unit library and evaluated based on a concatenation cost that describes the likelihood of the sequence of notes (based on a note-level LSTM) and a semantic relevance cost that describes the relationship between the two units in an embedding space (based on a DSSM).

For training these models we use the same dataset described in the previous section. However, in order to ensure that the model learns sequences and relationships that are musically appropriate we can only augment the dataset by transposing the pieces to different keys. Transposing does not compromise the original structure, pitch intervals, or rhythmic information within the data, however, the other transformations do affect these musical attributes and such transformations should not be applied for learning the parameters of these sequential models. However, it is possible to use the original unit library (including augmentations) when selecting units during generation.

Semantic Relevance

In both TTS and the previous musical reconstruction tests a target is provided. For generation tasks, however, the system must predict the next target based on the current sequential and contextual information that is available. In music, even if the content between two contiguous measures or phrases is different, there exist characteristics that suggest the two are not only related, but also likely to be adjacent to one another within the overall context of a musical score. We refer to this likelihood as the “semantic relevance” between two units.

This measure is obtained from a feature space learned using a DSSM. Though the underlying premise of the DSSM is similar to the DBN autencoder in that the objective is to learn good features in a compressed semantic space, the DSSM features, however, are derived in order to describe the relevance between two different units by specifically maximizing the posterior probability of consecutive units,

$P(u_n|u_{n-1})$, found in the training data. The same BOW features described in the previous section are used as input to the model. There are two hidden layers and the output layer describes the semantic feature vector used for computing the relevance. Each layer has 128 rectified linear units. The same softmax that was used for the autoencoder for computing loss is used for the DSSM. However, the loss is computed within vectors of the embedding space such that

$$-\log \prod_{(u_{n-1}, u_n)} P(\vec{u}_n | u_{n-1}) \quad (5)$$

where the vectors, \vec{u}_n and u_{n-1} , represent the 128 length embeddings of each unit derived from the parameters of the DSSM. Once the parameters are learned through gradient descent the model can be used to measure the relevance between any two units, U_1 and U_2 , using cosine similarity $sim(\vec{U}_1, \vec{U}_2)$ (see Equation 1).

The DSSM provides a meaningful measure between two units, however, it does not describe how to join the units (which one should come first). Similarly, the BOW representation of the input vector does not contain information that is relevant for making decisions regarding sequence. In order to optimally join two units a second measure is necessary.

Concatenation Cost

By using a unit library made up of original human compositions or improvisations, we can assume that the information within each unit is musically valid. In an attempt to ensure that the music remains valid after combining new units we employ a concatenation cost to describe the quality of the join between two units. This cost requires sequential information at a more fine grained level than the BOW-DSSM can provide.

We use a multi-layer LSTM to learn a note-to-note level model (akin to a character level language model). Each state in the model represents an individual note that is defined by its pitch and duration. This constitutes about a 3,000 note vocabulary. Using a one-hot encoding for the input, the model is trained to predict the next note, y_T , given a sequence, $\mathbf{x} = (x_1, \dots, x_T)$, of previously seen notes. During training, the output sequence, $\mathbf{y} = (y_1, \dots, y_T)$, of the network is such that $y_t = x_{t+1}$. Therefore, the predictive distribution of possible next notes, $Pr(x_{T+1}|\mathbf{x})$, is represented in the output vector, y_T . We use a sequence length of $T = 36$.

The aim of the concatenation cost is to compute a score evaluating the transition between the last note of the unit, u_{n-1, x_T} , and the first note of the unit, u_{n, y_T} . By using an LSTM it is possible to include additional context and note dependencies that exist further in the past than u_{n-1, x_T} . The cost between two units is computed as

$$C(u_{n-1}, u_n) = -\frac{1}{J} \sum_j \log Pr(x_j | \mathbf{x}_j) \quad (6)$$

where J is the number of notes in u_n , x_j is the j th note of u_n , and \mathbf{x}_j is the sequence of notes (with length T) immediately before x_j . Thus, for $j > 1$ and $j < T$, \mathbf{x}_j will include notes

from u_n and u_{n-1} and for $j \geq T$, \mathbf{x}_j will consist of notes entirely from u_n . In practice, however, the DSSM performs better than the note-level LSTM for predicting the next unit and we found that computing C with $J = 1$ provides the best performance. Therefore, the quality of the join is determined using only the first note of the unit in question (u_n).

The sequence length, $T = 36$, was chosen because it is roughly the average number of notes in four measures of music (from our dataset). Unlike the DSSM, which computes distances based on information from a fixed number of measures, the context provided to the LSTM is fixed in the number of notes. This means it may look more or less than four measures into the past. In the scenario in which there is less than 36 notes of available context the sequence is zero padded.

Ranking Units

A ranking process that combines the semantic relevance and concatenation cost is used to perform unit selection. Often times in music generation systems the music is not generated deterministically, but instead uses a stochastic process and samples from a distribution that is provided by the model. One reason for this is that note-level Markov chains or LSTMs may get “stuck” repeating the same note(s). Adding randomness to the procedure helps to prevent this. Here, we describe a deterministic method as this system is not as prone to repetitive behaviors. However, it is simple to apply stochastic decision processes to this system as the variance provided by sampling can be desirable if the goal is to obtain many different musical outputs from a single input seed.

The ranking process is performed in four steps:

1. Rank all units according to their semantic relevance with an input seed using the feature space learned by the DSSM.
2. Take the units whose semantic relevance ranks them in the top 5% and re-rank based on their concatenation cost with the input.
3. Re-rank the same top 5% based on their combined semantic relevance and concatenation ranks.
4. Select the unit with the highest combined rank.

By limiting the combined rank score to using only the top 5% we are creating a bias towards the semantic relevance. The decision to do this was motivated by findings from pilot listening tests in which it was found that a coherent melodic sequence relies more on the stylistic or semantic relatedness between two units than a smooth transition at the point of connection.

Evaluating the model

The model’s ability to choose good units can be evaluated using a ranking test. The task for the model is to predict the next unit given a never before seen four measures of music (from the held out test set). The prediction is made by ranking 50 candidates in which one is the truth and the other 49 are units randomly selected from the database. We repeat the experiments for musical units of different lengths including four, two, and one measures. The results are reported in

Table 2: Unit Ranking

Model	Unit length (measures)	Acc	Mean Rank @50
LSTM	4	17.2%	14.1
DSSM	4	33.2%	6.9
DSSM+LSTM	4	36.5%	5.9
LSTM	2	16.6%	14.8
DSSM	2	24.4%	10.3
DSSM+LSTM	2	28.0%	9.1
LSTM	1	16.1%	15.7
DSSM	1	19.7%	16.3
DSSM+LSTM	1	20.6%	13.9

the table below and they are based on the concatenation cost alone (LSTM), semantic relevance (DSSM), and the combined concatenation and semantic relevance using the selection process described above (DSSM+LSTM).

Discussion

As stated earlier the primary benefit of unit selection is being able to directly apply previously composed music. The challenge is stitching together units such that the musical results are stylistically appropriate and coherent. Another challenge in building unit selection systems is determining the optimal length of the unit. The goal is to use what has been seen before, yet have flexibility in what the system is capable of generating. The results of the ranking task may indicate that units of four measures have the best performance, yet these results do not provide any information describing the quality of the generated music.

Music inherently has a very high variance (especially when considering multiple genres). It may be that unit selection is too constraining and note-level control is necessary to create likeable music. Conversely, it may be that unit selection is sufficient and given an input sequence there may be multiple candidates within the unit database that are suitable for extending the sequence. In instances in which the ranking did not place the truth with the highest rank, we cannot assume that the selection is “wrong” because it may still be musically or stylistically valid. Given that the accuracies are not particularly high in the previous task an additional evaluation step is necessary to both evaluate the unit lengths and to confirm that the decisions made in selecting units are musically appropriate. In order to do this a subjective listening test is necessary.

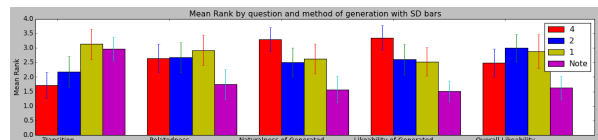


Figure 5: The mean rank and standard deviation for the different music generation systems using units of lengths 4, 2, and 1 measures and note level generation.

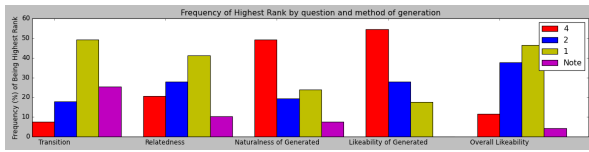


Figure 6: The frequency of being top ranked for the different music generation systems using units of lengths 4, 2, and 1 measures and note level generation. In both Figure 5 and 6 results are reported for each of the five hypotheses: 1) **Transition** – the naturalness of the transition between the first four measures (input seed) and last four measures (computer generated), 2) **Relatedness** – the stylistic or semantic relatedness between the first four measures and last four measures, 3) **Naturalness of Generated** – the naturalness of the last four measures only, 4) **Likeability of Generated** – the likeability of the last four measures only, and 5) **Overall Likeability** – the overall likeability of the entire eight measure sequence.

Subjective Evaluation

A subjective listening test was performed. Participants included 32 music experts in which a music expert is defined as an individual that has or is pursuing a higher level degree in music, a professional musician, or a music educator. Four systems were evaluated. Three of the systems employed unit selection using units of four, two, and one measures. The fourth system used the note-level LSTM to generate each note at a time.

The design of the test was inspired by subjective evaluations used by the TTS community. To create a sample each of the four systems was provided with the same input seed (retrieved from the held out dataset) and from this seed each then generated four additional measures of music. This process results in four eight-measure music sequences in which each has the same first four measures. The process was repeated 60 times using random four measure input seeds.

In TTS evaluations participants are asked to rate the quality of the synthesis based on naturalness and intelligibility (Stevens et al. 2005). In music performance systems the quality is typically evaluated using naturalness and likeability (Katayose et al. 2012). For a given listening sample, a participant is asked to listen to four eight-measure sequences (one for each system) and then are asked to rank the candidates within the sample according to questions pertaining to:

1. Naturalness of the transition between the first and second four measures.
2. Stylistic relatedness of the first and second four measures.
3. Naturalness of the last four measures.
4. Likeability of the last four measures.
5. Likeability of the entire eight measures.

Each participant was asked to evaluate 10 samples that were randomly selected from the original 60, thus, all participants listened to music generated by the same four systems, but the actual musical content and order randomly differed from

Table 3: Subjective Ranking

Variable	Best -> Worst
H1 - Transition Naturalness	1, N, 2, 4
H2 - Semantic Relatedness	1, 2, 4, N
H3 - Naturalness of Generated	4, 1, 2, N
H4 - Likeability of Generated	4, 2, 1, N
H5 - Overall Likeability	2, 1, 4, N

participant to participant. The tests were completed online with an average duration of roughly 80 minutes.

Results

Rank order tests provide ordinal data that emphasize the relative differences among the systems. The average rank was computed across all participants similarly to TTS-MOS tests. The percent of being top ranked was also computed. These are shown in Figures 5 and 6.

In order to test significance the non-parametric Friedman test for repeated measurements was used. The test evaluates the consistency of measurements (ranks) obtained in different ways (audio samples with varying input seeds). The null hypothesis states that random sampling would result in sums of the ranks for each music system similar to what is observed in the experiment. A Bonferoni post-hoc correction was used to correct the p-value for the five hypotheses (derived from the itemized question list described earlier).

For each hypothesis the Friedman test resulted in $p < .05$, thus, rejecting the null hypothesis. The sorted ranks for each of the generation system is described in Table 3.

Discussion

In H3 and H4 the participants were asked to evaluate the quality of the four generated measures alone (disregarding the seed). This means that the sequence resulting from the system that generates units of four measure durations are the unadulterated four measure segments that occurred in the original music. Given there was no computer generation or modification it is not surprising that the four measure system was ranked highest.

The note level generation performed well when it comes to evaluating the naturalness of the transition at the seams between the input seed and computer generated music. However, note level generation does not rank highly in the other categories. Our theory is that as the note-level LSTM accumulates error and gets further away from the original input seed the musical quality suffers. This behavior is greatly attenuated in a unit selection method assuming the units are pulled from human compositions.

The results indicate that there exists an optimal unit length that is greater than a single note and less than four measures. This ideal unit length appears to be one or two measures with a bias seemingly favoring one measure. However, to say for certain an additional study is necessary that can better narrow the difference between these two systems.

Conclusion

We present a method for music generation that utilizes unit selection. The selection process incorporates a score based on the semantic relevance between two units and a score based on the quality of the join at the point of concatenation. Two variables essential to the quality of the system are the breadth and size of the unit database and the unit length. An autoencoder was used to demonstrate the ability to reconstruct never before seen music by picking units out of a database. In the situation that an exact unit is not available the nearest neighbor computed within the embedded vector space is chosen. A subjective listening test was performed in order to evaluate the generated music using different unit durations. Music generated using units of one or two measure durations tended to be ranked higher according to overall likeability than units of four measures or note-level generation.

The system described in this paper generates monophonic melodies and currently does not address situations in which the melodies should conform to a provided harmonic context (chord progression) such as in improvisation. Plans for addressing this are included in future work. Additionally, unit selection may sometimes perform poorly if good units are not available. In such scenarios a hybrid approach that includes unit selection and note-level generation can be useful by allowing the system to take advantage of the structure within each unit whenever appropriate, yet, not restricting the system to the database. Such an approach is also planned for future work.

References

- Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.
- Black, A. W., and Taylor, P. A. 1997. Automatically clustering similar units for unit selection in speech synthesis.
- Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.
- Chordia, P.; Sastry, A.; and Şentürk, S. 2011. Predictive tabla modelling using variable-length markov and hidden markov models. *Journal of New Music Research* 40(2):105–118.
- Coca, A. E.; Romero, R. A.; and Zhao, L. 2011. Generation of composed musical structures through recurrent neural networks based on chaotic inspiration. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 3220–3226. IEEE.
- Collins, T.; Laney, R.; Willis, A.; and Garthwaite, P. H. 2016. Developing and evaluating computational models of musical style. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 30(01):16–43.
- Conkie, A.; Beutnagel, M. C.; Syrdal, A. K.; and Brown, P. E. 2000. Preselection of candidate units in a unit selection-based text-to-speech synthesis system. In *Proc. ICSLP, Beijing*.
- Cope, D. 1999. One approach to musical intelligence. *IEEE Intelligent systems and their applications* 14(3):21–25.
- Eck, D., and Schmidhuber, J. 2002. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, 747–756. IEEE.
- Franklin, J. A. 2006. Recurrent neural networks for music computation. *INFORMS Journal on Computing* 18(3):321–338.
- Goel, K.; Vohra, R.; and Sahoo, J. 2014. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn. In *Artificial Neural Networks and Machine Learning–ICANN 2014*. Springer. 217–224.
- Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, 2333–2338. ACM.
- Hunt, A. J., and Black, A. W. 1996. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, 373–376. IEEE.
- Katayose, H.; Hashida, M.; De Poli, G.; and Hirata, K. 2012. On evaluating systems for generating expressive music performance: the recon experience. *Journal of New Music Research* 41(4):299–310.
- Lerdahl, F., and Jackendoff, R. 1987. A generative theory of tonal music.
- Lerdahl, F. 1992. Cognitive constraints on compositional systems. *Contemporary Music Review* 6(2):97–121.
- McCormack, J. 1996. Grammar based music composition. *Complex systems* 96:321–336.
- Norgaard, M. 2014. How jazz musicians improvise. *Music Perception: An Interdisciplinary Journal* 31(3):271–287.
- Pachet, F., and Roy, P. 2011. Markov constraints: steerable generation of markov sequences. *Constraints* 16(2):148–172.
- Papadopoulos, G., and Wiggins, G. 1999. Ai methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, 110–117. Edinburgh, UK.
- Pressing, J. 1988. Improvisation: methods and models. *John A. Sloboda (Hg.): Generative processes in music, Oxford* 129–178.
- Simon, I.; Morris, D.; and Basu, S. 2008. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 725–734. ACM.
- Stevens, C.; Lees, N.; Vonwiller, J.; and Burnham, D. 2005. Online experimental methods to evaluate text-to-speech (tts) synthesis: effects of voice gender and signal quality on intelligibility, naturalness and preference. *Computer speech & language* 19(2):129–146.
- van den Oord, A.; Kalchbrenner, N.; Vinyals, O.; Espeholt, L.; Graves, A.; and Kavukcuoglu, K. 2016. Conditional image generation with pixelcnn decoders. *CoRR* abs/1606.05328.
- Wang, C.-i., and Dubnov, S. 2014. Guided music synthesis with variable markov oracle. In *3rd International Workshop on Musical Metacreation, Raleigh, NC, USA*.
- Zen, H.; Tokuda, K.; and Black, A. W. 2009. Statistical parametric speech synthesis. *Speech Communication* 51(11):1039–1064.

A Pig, an Angel and a Cactus Walk Into a Blender: A Descriptive Approach to Visual Blending

João M. Cunha, João Gonçalves, Pedro Martins, Penousal Machado, Amílcar Cardoso

CISUC, Department of Informatics Engineering

University of Coimbra

{jmacunha,jcgonc,pjmm,machado,amilcar}@dei.uc.pt

Abstract

A descriptive approach for automatic generation of visual blends is presented. The implemented system, the *Blender*, is composed of two components: the *Mapper* and the *Visual Blender*. The approach uses structured visual representations along with sets of visual relations which describe how the elements – in which the visual representation can be decomposed – relate among each other. Our system is a hybrid blender, as the blending process starts at the *Mapper* (conceptual level) and ends at the *Visual Blender* (visual representation level). The experimental results show that the *Blender* is able to create analogies from input mental spaces and produce well-composed blends, which follow the rules imposed by its base-analogy and its relations. The resulting blends are visually interesting and some can be considered as unexpected.

Introduction

Conceptual Blending (CB) theory is a cognitive framework proposed by Fauconnier and Turner (2002) as an attempt to explain the creation of meaning and insight. CB consists in integrating two or more mental spaces in order to produce a new one, the blend(ed) space. Here, mental space means a temporary knowledge structure created for the purpose of local understanding (Fauconnier 1994).

Visual blending, which draws inspiration from CB theory, is a relatively common technique used in Computational Creativity to generate creative artefacts in the visual domain. While some of the works are explicitly based on Conceptual Blending theory, as blending occurs at a conceptual level, other approaches generate blends only at a representation/instance level by means of, for example, image processing techniques.

We present a system for automatic generation of visual blends (*Blender*), which is divided into two different parts: the *Mapper* and the *Visual Blender*. We follow a descriptive approach in which a visual representation for a given concept is constructed as a well-structured object (from here onwards when we use the term representation we are referring to visual representations). The object can contain other objects and has a list of descriptive relations, which

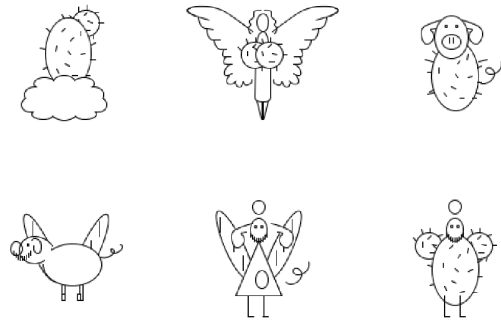


Figure 1: Examples of produced blends.

describe how the object relates to others. The relations describe how the representation is constructed (example: *part A* inside *part B*). In our opinion, this approach allows an easier blending process and contributes to the overall sense of cohesion among the parts.

Our system can be seen as a hybrid blender, as the blending process starts at the conceptual level (which occurs in the *Mapper*) and only ends at the visual representation level (which occurs in the *Visual Blender*). We use an evolutionary engine based on a Genetic Algorithm, in which each population corresponds to a different analogy and each individual is a visual blend. The evolution is guided by a fitness function that assesses the quality of each blend based on the satisfied relations. In the scope of this work, the focus is given to the *Visual Blender*.

Related Work

In terms of the type of rendering, current computational approaches to visual blending can be divided into two groups: the ones which attempt to blend pictures or photorealistic renderings; and the ones that focus on non-photorealistic representations, such as pictograms or icons.

The Boat-House Visual Blending Experience (Pereira and Cardoso 2002) is, to the best of our knowledge, one of the earliest attempts to computationally produce visual blends. The work was motivated by the need to interpret and visualize blends produced by a preliminary version of the Divago framework, which is one of the first artificial creative

systems based on CB theory (Pereira 2007). In addition to a declarative description of the concepts via rules and concept maps (i.e., graphs representing binary relations between concepts), Pereira and Cardoso also considered a domain of instances, which were drawn using a Logo-like programming language. To test the system, the authors performed several experiments with the *house* and *boat* blend (Goguen 1999) considering different instances for the input spaces.

Ribeiro et al. (2003) explored the use of the Divago framework in procedural content generation. In this work, the role of Divago was to produce novel creatures at a conceptual level from a set of existing ones. Then, a 3D interpreter was used to visualize the objects. The interpreter was able to convert concept maps from Divago, representing creatures, into Wavefront OBJ files that could be rendered afterwards.

Steinbrück (2013) introduced a framework that formalises the process of CB while applying it to the visual domain. The framework is composed of five modules that combine image processing techniques with gathering semantic knowledge about the concept depicted in an image with the help of ontologies. Elements of the image are replaced with other unexpected elements of similar shape (for example, round medical tablets are replaced with pictures of a globe).

Confalonieri et al. (2015) proposed a discursive approach to evaluate the quality of blends (although there is no evidence of an implementation). The main idea was to use Lakatosian argumentative dialogue (Lakatos 1976) to iteratively construct valuable and novel blends as opposed to a strictly combinatorial approach. To exemplify the argumentative approach, the authors focused on icon design by introducing a semiotic system for modelling computer icons. Since icons can be considered as a combination of signs that can convey multiple intended meanings to the icon, Confalonieri et al. proposed argumentation to evaluate and refine the quality of the icons.

Xiao and Linkola (2015) proposed Vismantic, a semi-automatic system aimed at producing visual compositions to express specific meanings, namely the ones of abstract concepts. Their system is based on three binary image operations (juxtaposition, replacement and fusion), which are the basic operations to represent visual metaphors (Phillips and McQuarrie 2004). For example, Vismantic represents the slogan *Electricity is green* as an image of an electric light bulb where the wire filament and screw base are fused with an image of green leaves. The selection of images as well as the application of the visual operations require user's intervention.

Correia et al. (2016) proposed X-Faces, which can be seen as a data augmentation technique to autonomously generate new faces out of existing ones. Elementary parts of the faces, such as eyes, nose or mouth, are recombined by means of evolutionary algorithms and computer vision techniques. The X-Faces framework generates unexpected, yet realistic, faces by exploring the shortcomings and vulnerabilities of computational face detectors to promote the evolution of faces that are not recognised as such by these systems.

Recent works such as DeepStyle (Gatys, Ecker, and Bethge 2015) can also be seen as a form of visual blend-

ing. DeepStyle is based on a deep neural network that has the ability to separate image content from certain aspects of style, allowing to recombine the content of an arbitrary image with a given rendering style (style transfer). The system is known for mimicking features of different painting styles.

Several other authors have seen the potential of deep neural networks for tasks related to visual blending (Berov and Kuhnberger 2016; McCaig, DiPaola, and Gabora 2016; Heath and Ventura 2016). For instance, Berov and Kühnberger (2016) proposed a computational model of visual hallucination based on deep neural networks. To some extent, the creations of this system can be seen as visual blends.

The approach

Having the organization of mental spaces as an inspiration, we follow a similar approach to structure the construction of the visual representations, which are considered as a group of several parts / elements. By focusing on the parts instead of the whole, there is something extra that stands out: not only is given importance to the parts but the representation ceases to be a whole and starts to be seen as parts related to each other. As our goal is to produce visual results, these relations have a visual descriptive nature (i.e. the nature of the relation between two elements is either related to their relative position or to their visual qualities). This allows the generation of visual blends, guided and evaluated by criteria imposed by the relations present in the base-representations (see Fig.3) used in the visual blend production.

In addition, by using a representation style that consists of basic shapes, we reduce the concept to its simplest form, maintaining its most important features and thus, hopefully, capturing its essence (a similar process can be seen in Picasso's *The Bull*, a set of eleven lithographs produced in 1945). As such, our approach can be classified as belonging to the group of non-photorealistic visual blending. This simplification of concepts has as inspiration several attempts to produce a universal language, understandable by everyone – such as the pictographic ISOTYPE by Otto Neurath (1936) or the symbolic Blissymbolics by Charles Bliss (1965).

As already mentioned, our main idea is centered on the fact that the construction of a visual representation for a given concept can be approached in a structured way. Each representation is associated with a list of descriptive relations (e.g.: *part A* below *part B*), which describes how the representation is constructed. Due to this, a visual blend between two representations is not simply a replacement of parts but its quality is assessed based on the number of relations that are respected. This gives much more flexibility to the construction of representations by presenting a version of it and also allowing the generation of similar ones, if needed.

The initial idea involved only a representation for each concept. However, a given concept has several possible visual representations (e.g. there are several possible ways of visually representing the concept *car*), which means that only using one would make the system very limited.

In order to avoid biased results, we decided to use several versions for each concept. Each visual representation can



Figure 2: On the left is a the representation drawn with the elements identified; On the right is the result of the conversion into fully scalable vector graphic.

be different (varying in terms of style, complexity, number of characteristics and even chosen perspective) and thus also having a different set of visual relations among the parts.

In comparison to the systems described in the previous Section, we follow a different approach to the generation of visual blends by implementing a hybrid system and giving great importance to the parts and their relations – such tends to be overlooked by the majority of the reviewed works in which an unguided replacement of parts often leads to a lack of cohesion among them. This approach allows us not only to assess the quality of the blends and guide evolution but also to easily generate similar (and also valid) blends based on a set of relations.

Collecting data

The initial phase of the project consisted in a process of data collection. Firstly, a list of possible concepts was produced by collecting concepts already used in the conceptual blending field of research. From this list, three concepts were selected based on their characteristics: *angel* (human-like), *pig* (animal) and *cactus* (plant) – collected from Keane and Costello (2001). The goal of this phase was to collect visual representations for these concepts. An enquiry to collect the desired data was designed, which was composed of five tasks:

- T1** Collection of visual representations for the selected concepts;
- T2** Identification of the representational elements;
- T3** Description of the relations among the identified elements;
- T4** Identification of the prototypical elements – i.e. the element(s) that most identify a given concept (Johnson 1985). For instance, for the concept *pig* most participants considered *nose* and *tail* as the prototypical elements;
- T5** Collection of visual blends for the selected concepts.

The data was collected from nine participants who were asked to complete the required tasks. In the first task (T1), the participants were asked to draw a representation for each concept avoiding unnecessary complexity but still representing the most important elements of the concept. In order to achieve intelligible and relatively simple representations, the participants were suggested to use primitives such as lines, ellipses, triangles and quadrilaterals as the basis for their

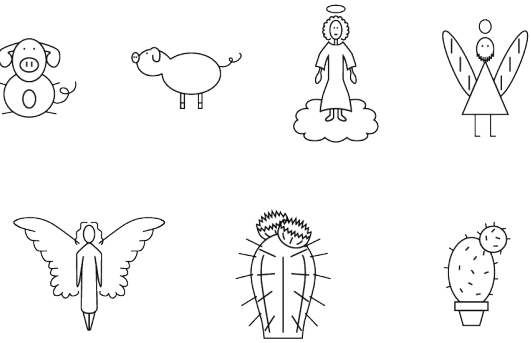


Figure 3: Representations used as a base.

drawings. After completing the first version, a second one was requested. The reason for two versions was to promote diversity.

In the second task (T2), the participants identified the elements drawn using their own terms (for example, for the concept *angel* some of the identified elements were *head*, *halo*, *legs*).

After completing the previous task, the participants were asked to identify the relations among elements that they considered as being essential (T3). These relations were not only related to the conceptual space but also (and mostly) to the representation. In order to help the participants, a list of relations was provided. Despite being told that the list was only to be considered as an example and not to be seen as closed, all the participants used the relations provided – this ensured the semantic sharing between participants. Some participants suggested other relations that were not on the list – these contributions were well-received.

The identified relations are dependent on the author's interpretation of the concept, which can be divided into two levels. The first level is related to how the author interprets the connections among the concepts of the parts at a conceptual level (for example *car*, *wheel* or *trunk*). The second level is related to the visual representation being considered: different visual representations may have different relations among the same parts (this can be caused, for example, by the change of perspective or style) – e.g. the different positioning of the head in the two *pig* representations in Fig.3.

Task four (T4) consisted in identifying the prototypical parts of the representations – the parts which most identify the concept (Johnson 1985). These will be used for interpreting the results obtained and for posterior developments.

In the last task of the enquiry (T5), the participants were asked to draw representations for the blends between the three concepts. As a blend between two concepts can be interpreted and posteriorly represented in different ways (e.g. just at a naming level a blend between *pig* and *cactus* can be differently interpreted depending on its name being *pig-cactus* or *cactus-pig*). For this reason, the participants were asked to draw one or more visual representations for the blend. These visual representations were later used for comparing with the results obtained with the *Visual Blender*.

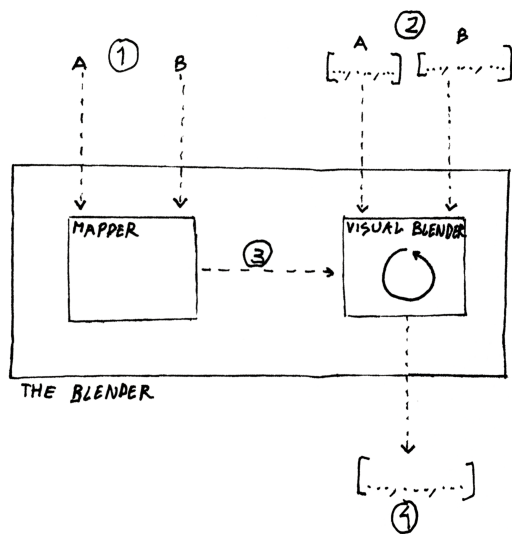


Figure 4: Structure of the implemented Blender. The Blender consists of a Mapper and a visual Blender. The figure also shows the input spaces (1), the visual representations and list of relations (2), the produced analogies (3) and the produced blends (4).

Post-enquiry

After the conduction of the enquiry, the data was treated in order to be used by the *Visual Blender*. Firstly, the representations collected for each of the concepts were converted into fully scalable vector graphics (see Fig. 2) and prepared to be used as base visual representations (see Fig.3) for the *Visual Blender* (using layer naming according to the data collected for each representation – each layer was named after its identified part). In addition to this, the relations among parts were formatted to be used as input together with their corresponding representation.

The Visual Blender

As already mentioned, the *Blender* has two different components: the *Mapper* and the *Visual Blender* (see Fig.4). The *Mapper* receives two input spaces (represented as 1 in Fig.4), one referring to *concept A* and the other one to *concept B*. It produces analogies (3 in Fig.4) that are afterwards used by the *Visual Blender* component. The *Visual Blender* also receives visual representations and corresponding list of relations among parts (2 in Fig.4) that are used as a base and data for producing the visual blends (4 in Fig.4).

As this paper is focused on the *Visual Blender* component, the *Mapper* is only briefly described (subsection *Generating the blends: structural mapping*). Despite being related, the two components have different implementation details (e.g. object structure).

Generating the blends: structural mapping

In Conceptual Blending theory, after the selection of input spaces, the subsequent step is to perform a partial matching

between elements of the given mental spaces. This can be seen as establishing an analogy between the two inputs. Our input spaces are in the form of semantic maps composed of N_c concepts and N_t triples, with $N_t, N_c \in \mathbb{N}$. The triples are in the form $\langle concept_0, relation, concept_1 \rangle$. Each concept corresponds to a vertex in a generic graph and the relation represents a directed edge connecting both concepts.

The *Mapper* iterates through all possible root mappings, each composed of two distinct concepts taken from the input spaces. This means that there is a total of $\binom{N_c}{2}$ iterations. Then, the algorithm extracts two isomorphic sub-graphs from the larger input space. The two sub-graphs are split in two sets of vertices *A* (left) and *B* (right). The structural isomorphism is defined by the sequence of relation types (pw, isa,...) found in both sub-graphs.

Starting at the root mapping defined by two (left and right) concepts, the isomorphic sub-graphs are extracted from the larger semantic structure (the input spaces) by executing two synchronised expansions of nearby concepts at increasingly depths. The first expansion starts from the left concept and the second from the right concept. The left expansion is done recursively in the form of a depth first expansion and the right as a breadth first expansion. The synchronisation is controlled by two mechanisms:

1. the depth of the expansion, which is related to the number of relations reached by each expansion, starting at either concept from the root mapping;
2. the label used for selecting the same relation to be expanded next in both sub-graphs.

Both left (depth) and right (breadth) expansions are always synchronized at the same level of deepness (first mechanism above).

While expanding, the algorithm stores additional associations between each matched relations and the corresponding concept which was reached through that relation. In reality, what is likely to happen is to occur a multitude of isomorphisms. In that case, the algorithm will store various mappings from any given concept to multiple different concepts, as long as the same concepts were reached from a previous concept with the same relation. In the end, each isomorphism and corresponding set of concept mappings gives rise to an analogy. The output of the *Mapper* component is a list of analogies with the greatest number of mappings.

Generating the blends: construction and relations

The *Visual Blender* component uses structured base-representations (of the input concepts) along with their set of relations among parts to produce visual blends based on analogies (mappings) produced by the *Mapper* component.

The way of structuring the representations is based on the *Syntactic decomposition of graphic representations* proposed by von Engelhardt (2002) in which a composite graphic object consists of: a graphic space (occupied by the object); a set of graphic objects (which may also be composite graphic objects); and a set of graphic relations (which may be *object-to-space* and/or *object-to-object*).

The objects store several attributes: name, shape, position relative to the father-object (which has the object in the

set of graphic objects), the set of relations to other objects and the set of child-objects. By having such a structure, the complexity of blending two base representations is reduced, as it facilitates object exchange and recursive changing (by moving an object, the child-objects are also easily moved).

A relation between two objects consists of: the object *A*, the object *B* and the type of relation (*above*, *lowerPart*, *inside*, ...) – e.g. *eye (A) inside head (B)*.

Generating the blends: visual blending

The *Visual Blender* receives the analogies between two given concepts produced by the *Mapper* component and the blend step occurs during the production of the visual representation – differently from what happens in *The Boat-House Visual Blending Experience* (Pereira and Cardoso 2002), in which the blends are merely interpreted at the visual representation level.

The part of the blending process that occurs at the *Visual Blender* produces visual representations as output and consists of five steps:

- S1** An analogy is selected from the set of analogies provided by the *Mapper*;
- S2** One of the concepts (either *A* or *B*) is chosen as a base (consider *A* as the chosen one, as an example);
- S3** A visual representation (*rA*) is chosen for the concept *A* and a visual representation (*rB*) is chosen for the concept *B*;
- S4** Parts of *rA* are replaced by parts of *rB* based on the analogy. For each mapping of the analogy – consider for example *leg* of *A* corresponds to *arm* of *B* – the following steps occur:
 - S4.1** The parts from *rA* that correspond to the element in the mapping (e.g. *leg*) are searched using the names of the objects. In the current example, the parts found could be *left_leg* (*left_* is a prefix), *right_leg_1* (*right_* is a prefix and *_1* a suffix) or even *leftfront_leg*;
 - S4.2** For each of the found parts in S4.1, a matching part is searched in *rB* using the names of the objects. This search firstly looks for objects that match the full name, including the prefix and suffix (e.g. *right_arm_1*) and, if none is found, searches only using the name in the mapping (e.g. *arm*). It avoids plural objects (e.g. *arms*). If no part is found, it proceeds to step S4.4;
 - S4.3** The found part (*pA*) of *rA* is replaced by the matching part (*pB*) of *rB*, updating the relative positions of *pB* and its child-objects, and relations (i.e. relations that used to belong to *pA* now point to *pB*);
 - S4.4** A process of Composition occurs (see examples in Fig.5 – the *tail* and the *belly / round shape* in the triangular *body* are obtain using composition). For each of the matching parts from *rB* (even if the replacement does not occur) a search is done for parts from *rB* that have a relation with *pB* (for example, a found part could be *hand*). It only accepts a part if *rA* does not have a part with the same name and if the analogy used does not have a mapping for it. If a found part matches these criteria, a composition can occur by copying the part



Figure 5: The “face expressions” of the angel-pigs – given the same or similar rules, the produced results are still quite diverse. The *tail* and the *belly / round shape* in the triangular *body* are obtain through a process of composition (S4.4).

to *rA* (in our example, depending on either the replacement in Step S4.3 occurred or not, *rA* would have either *hand* related to *arm* or to *leg*, respectively);

- S5** The *rA* resulting from the previous steps is checked for inconsistencies (both in terms of relative positioning and obsolete relations – which can happen if an object does not exist anymore due to a replacement);

After generating a representation, the similarity to the base representations (*rA* and *rB*) is assessed to avoid producing representations visually equal to them. This assessment is done by using a Root Mean Square Error (RMSE) measure that checks the similarity on a pixel-by-pixel basis.

Evolutionary Engine

The main goal of the *Visual Blender* component is to produce and evolve possible visual blends based on the analogies produced by the *Mapper*. In order to achieve this and promote diversity while respecting each analogy, an evolutionary engine was implemented. This engine is based on a Genetic Algorithm (GA) using several populations (each corresponding to a different analogy), in which each individual is a visual blend.

In order to guide evolution, we adopt a fitness function that assesses how well the the existing relations are respected. Some of the relations, e.g. the relation *above*, have a binary assessment – either 0, when the relation is not respected, or 1 when it is respected. Others yield a value between 0 and 1 depending on how respected it is – e.g. the relation *inside* calculates the number of points that are inside and returns $\frac{\#PointsInside}{total\#Points}$.

The fitness function for a given visual blend *b* is as follows:

$$f(b) = \frac{\sum_{i=1}^{\#R(b)} v(r_i(b))}{\#R(b)}, \quad (1)$$

where $\#R(b)$ denotes the number of relations present in *b* and *v* is the function with values in $[0, 1]$ that indicates how much a relation *r* is respected (0 – not respected at all, 1 – fully respected).

The evolutionary engine includes five tasks which are performed in each generation for each population:

- T1** Produce more individuals when the population size is below the maximum size;
- T2** Store the best individual to avoid losing it (elitism);
- T3** Mutate the individuals of the population. For each individual, each object can be mutated by changing its position. This change also affects its child-objects;
- T4** Recombine the individuals: the parents are chosen using tournament selection (with size 2) and a *N-point crossover* is used to produce the children. In order to avoid the generation of invalid individuals, the crossover only occurs between chromosomes (objects) with the same name (e.g. a *head* is only exchanged with a *head*). If this rule was not used, it would lead to the production of descendants that would not respect the analogy followed by the population;
- T5** Removal of identical individuals in order to increase variability.

In the experiments reported in this paper the mutation probability was set to 0.05, per gene, and the recombination probability to 0.2, per individual. These values were established empirically in preliminary runs.

Results and discussion

In this section we present and discuss the experimental results. We begin with a general analysis. Afterwards, we analyse the resulting visual representations comparing them with the data collected in the initial enquiry. Then, we analyse the quality of the produced blends by presenting the results of a final enquiry focused on perception.

Overall, the analysis of the experimental results indicates that the implemented blender is able to produce sets of blends with great variability (see Fig.5 for an example of the results obtained for the same analogy and the same relations) and unexpected features, while respecting the analogy. The evolutionary engine is capable of evolving the blends towards a higher number of satisfied relations. This is verifiable in numerical terms, through the analysis of the evolution of fitness, and also through the visual assessment of the results. Figure 6 illustrates the evolution of a blend: the *legs* and *tail* are iteratively moved towards the *body* in order to increase the degree of satisfaction of the relations.

We can also observe that the system tends to produce blends in which few parts are exchanged between concepts. This can be explained as follows: when the number of parts increases the difficulty of (randomly) producing a blend with adequate fitness drastically decreases. As such, blends with fewer exchanges of parts, thus closer to base representation (in which all the relations are satisfied), tend to become dominant during the initial generations of the evolutionary runs. We consider that a significantly higher number of runs would be necessary to produce blends with more exchanges. Furthermore, valuing the exchange of parts, through the modification of the fitness function, may also be advisable for promoting the emergence of such blends.

As the blends are being produced as a visual representation which works as a whole as well as a set of individual



Figure 6: Evolution of a blend: the *legs* and *tail* come closer to the *body*, guided by the fitness function.

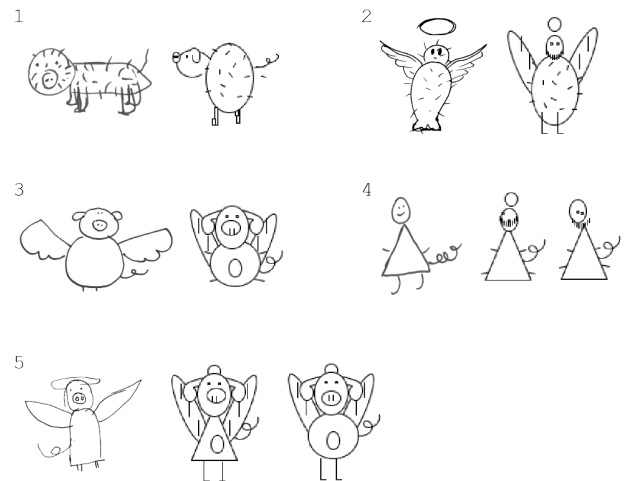


Figure 7: Comparison between hand-drawn blends and blends generated by the implemented *Blender*, organised by groups: group 1 corresponds to *pig-cactus* blends; 2 corresponds to *angel-cactus*; groups 3-5 correspond to *pig-angel* (the figure on the left of each group is the hand-drawn blend).

parts, the Principle of Integration is being respected by design – from the Optimality Principles presented by Fauconnier and Turner (1998).

Comparison with user-drawn blends

During the initial phase of the project, we conducted a task of collecting visual blends drawn by the participants. A total of 39 drawn blends were collected, from which 14 correspond to the blend between *cactus* and *angel*, 12 correspond to the blend between *cactus* and *pig* and 13 correspond to the blend between *pig* and *cactus*. The implemented blender was able to produce visual blends similar to the ones drawn by the participants (see some examples in Fig. 7). After analysing the produced blends, the following results were obtained:

- 23 from the 39 drawn blends (DB) were produced by our *Blender*;
- 2 are not possible to be produced due to inconsistencies (e.g. one drawn blend from *angel-pig* used a mapping from *wing-tail* and at the same time maintained the *wings*);
- 6 were not able to be produced in the current version due

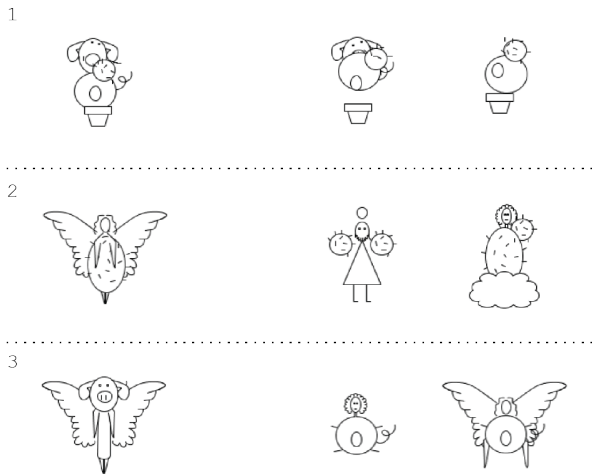


Figure 8: Examples of the visual blends presented in the second enquiry. On the left are the “good” blends (one for each) and on the right are the “bad” blends (1 corresponds to *cactus-pig*, 2 to *angel-cactus* and 3 to *angel-pig*).

to mappings that were not produced by the *Mapper* (e.g. *head* from *angel* with *body* from *cactus*);

- 5 were not able to be produced because not all of the collected drawn representations were used in the experiments.

According to the aforementioned results, the implemented *Blender* is not only able to produce blends that are coherent with the ones drawn by participants but is also able to produce novel blends that no participant drew, showing creative behaviour.

Evaluating perception

In order to assess if the produced blends could be correctly perceived, a second enquiry was conducted. The main goal was to evaluate whether or not the participant could identify the input spaces used for each blend (i.e. if it was possible to identify *pig* and *cactus* in a blend produced for *pig-cactus*). This is related to the Unpacking Principle (Fauconnier and Turner 1998).

In the first enquiry, the fourth task (T4) consisted in collecting the prototypical parts for each concept – these are the parts that most identify the concept (e.g. *wing* for *angel*). We used the data collected for producing the second enquiry. For each blend (*angel-pig*, *cactus-pig* or *angel-cactus*), four visual blends were selected (two considered “good” and two considered “bad”, see Fig. 8). The quality evaluation (“bad” or “good”) was based on two criteria: fitness of the individual and presence or legibility of the prototypical parts (i.e. a “good” exemplar is an individual with the prototypical parts clearly identifiable; a “bad” exemplar is an individual with fewer prototypical parts or these are not clearly identifiable).

A total of 12 visual blends were used and the enquiry was conducted to 30 participants. Each visual blend was

Table 1: Number of correct names given (input spaces’ names) for each of the blends (percentage of answers).

		0	1	2
cactus-pig	Good	20	50	30
	Bad	50	50	0
angel-pig	Good	10	20	70
	Bad	40	50	10
angel-cactus	Good	0	60	40
	Bad	10	80	10

Table 2: Number of correct names given (input spaces’ names) for each of the blends (number of answers).

		# R.	0	1	2
cactus-pig	Good	1	1	2	2
		2	1	3	1
	Bad	3	1	4	0
		4	4	1	0
angel-pig	Good	5	0	1	4
		6	1	1	3
	Bad	7	2	3	0
		8	2	2	1
angel-cactus	Good	9	0	4	1
		10	0	2	3
	Bad	11	0	5	0
		12	1	3	1

tested by 5 participants. In order to minimise the biasing of the results, each participant evaluated two visual representations (one “bad” and one “good”) of different blends (e.g. when the first was of *cactus-pig*, the second could only be of *angel-pig* or *angel-cactus*). The “bad” blends were evaluated first to further minimise the biasing.

The results (Table 1 and Table 2), clearly show that the “good” blends were easier to be correctly named (the percentage of total correct naming is always higher for the “good” examples; the percentage of total incorrect naming is always higher for the “bad” blends). In addition to this, the names of the input spaces were also easier to be identified in some of the representations than in others (e.g. the “good” blends for *angel-pig* received more totally correct answers than the rest of the blends, as shown in Table 2).

Overall, the majority of the participants could identify at least one of the input spaces for the “good” exemplars of visual blends. Even though some of the participants could not correctly name both of the input spaces, the answers given were somehow related to the correct ones (e.g. the names given for the input spaces in the first “bad” blend of 3 in Fig. 8 were often *pig* and *lady/woman*, instead of *pig* and *angel* – this is due to the fact that no *halo* nor *wings* are presented).

Conclusions and future work

We presented a descriptive approach for automatic generation of visual blends. The approach uses structured representations along with sets of visual relations which describe how the parts – in which the visual representation can be decomposed – relate among each other. The experimental results demonstrate the ability of the *Blender* to produce analogies from input mental spaces and generate a wide variety of visual blends based on them. The *Visual Blender* component, in addition to fulfilling its purpose, is able to produce interesting and unexpected blends. Future enhancements to the proposed approach include:

- (i) exploring an island approach in which exchange of individuals from different analogies may occur if they respect the analogy of the destination population;
- (ii) exploring the role of the user (guided evolution), by allowing the selection of individuals to evolve;
- (iii) considering Optimality Principles in the assessment of fitness (e.g. how many parts are exchanged) and exploring which of them may be useful or needed – something discussed by Martins et al. (2016);
- (iv) using relations such as *biggerThan* or *smallerThan* to explore style changing (e.g. the style of the produced blends will be affected if a base visual representation has head biggerThan body);
- (v) exploring context in the production of blends (e.g. stars surrounding the angel).

References

- Berov, L., and Kuhnberger, K.-U. 2016. Visual hallucination for computational creation. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Bliss, C. K. 1965. *Semantography (Blissymbolics): A Logical Writing for an illogical World*. Semantography Blissymbolics Publ.
- Confalonieri, R.; Corneli, J.; Pease, A.; Plaza, E.; and Schorlemmer, M. 2015. Using argumentation to evaluate concept blends in combinatorial creativity. In *Proc. of the Sixth Int. Conf. on Computational Creativity*, 174–181.
- Correia, J.; Martins, T.; Martins, P.; and Machado, P. 2016. X-faces: The exploit is out there. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Fauconnier, G., and Turner, M. 1998. Conceptual integration networks. *Cognitive Science* 22(2):133–187.
- Fauconnier, G., and Turner, M. 2002. *The Way We Think*. New York: Basic Books.
- Fauconnier, G. 1994. *Mental Spaces: Aspects of Meaning Construction in Natural Language*. New York: Cambridge University Press.
- Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.
- Goguen, J. 1999. An introduction to algebraic semiotics, with applications to user interface design. In *Lecture Notes in Artificial Intelligence*, volume Computation for Metaphor, Analogy and Agents, 242–291. Springer.
- Heath, D., and Ventura, D. 2016. Before a computer can draw, it must first learn to see. In *Proceedings of the 7th International Conference on Computational Creativity*, page to appear.
- Johnson, R. 1985. Prototype theory, cognitive linguistics and pedagogical grammar. *Working Papers in Linguistics and Language Training* 8:12–24.
- Keane, M. T., and Costello, F. J. 2001. Setting limits on analogy: Why conceptual combination is not structural alignment. In Gentner, D.; Holyoak, K.; and Kokinov, B., eds., *The Analogical Mind: A Cognitive Science Perspective*. Cambridge, MASS: MIT Press.
- Lakatos, I. 1976. *Proofs and refutations: the logic of mathematical discovery*. Cambridge University Press.
- Martins, P.; Pollak, S.; Urbancic, T.; and Cardoso, A. 2016. Optimality principles in computational approaches to conceptual blending: Do we need them (at) all? In *Proceedings of the Seventh International Conference on Computational Creativity*.
- McCaig, G.; DiPaola, S.; and Gabora, L. 2016. Deep convolutional networks as models of generalization and blending within visual creativity. *arXiv preprint arXiv:1610.02478*.
- Neurath, O. 1936. *International Picture Language. The First Rules of Isotype... With Isotype Pictures*. Kegan Paul & Company.
- Pereira, F. C., and Cardoso, A. 2002. The boat-house visual blending experience. In *Proceedings of the Symposium for Creativity in Arts and Science of AISB 2002*.
- Pereira, F. C. 2007. *Creativity and Artificial Intelligence: A Conceptual Blending Approach*. Berlin: Mouton de Gruyter.
- Phillips, B. J., and McQuarrie, E. F. 2004. Beyond visual metaphor: A new typology of visual rhetoric in advertising. *Marketing theory* 4(1-2):113–136.
- Ribeiro, P.; Pereira, F. C.; Marques, B.; Leitao, B.; and Cardoso, A. 2003. A model for creativity in creature generation. In *4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*.
- Steinbrück, A. 2013. *Conceptual blending for the visual domain*. Ph.D. Dissertation, Masters thesis, University of Amsterdam.
- von Engelhardt, J. 2002. *The language of graphics: A framework for the analysis of syntax and meaning in maps, charts and diagrams*. Yuri Engelhardt.
- Xiao, P., and Linkola, S. 2015. Vismantic: Meaning-making with images. In *Proceedings of the 6th Int. Conference on Computational Creativity, ICC-15*.

Distributed Musical Decision-making in an Ensemble of Musebots: Dramatic Changes and Endings

Arne Eigenfeldt

School for the
Contemporary Arts
Simon Fraser University
Vancouver, Canada
arne_e@sfu.ca

Oliver Bown

Art and Design
University of
New South Wales
Sydney, Australia
o.bown@unsw.edu.au

Andrew R. Brown

Queensland
College of Art
Griffith University
Brisbane, Australia
andrew.r.brown@griffith.edu.au

Toby Gifford

Sensilab
Monash University
Melbourne, Australia
toby.gifford@monash.edu

Abstract

A musebot is defined as a piece of software that autonomously creates music and collaborates in real time with other musebots. The specification was released early in 2015, and several developers have contributed musebots to ensembles that have been presented in North America, Australia, and Europe. This paper describes a recent code jam between the authors that resulted in four musebots co-creating a musical structure that included negotiated dynamic changes and a negotiated ending. Outcomes reported here include a demonstration of the protocol's effectiveness across different programming environments, the establishment of a parsimonious set of parameters for effective musical interaction between the musebots, and strategies for coordination of episodic structure and conclusion.

Introduction

Musebots are pieces of software that autonomously create music collaboratively with other musebots. A defining goal of the musebot project (Bown et al. 2015) is to establish a creative platform for experimenting with musical autonomy, open to people developing cutting-edge music intelligence, or simply exploring the creative potential of generative processes in music.

A larger and longer-term goal for the project has been a sharing of ideas about musebot programming, as well as the sharing of code. There already exists substantial research into Musical Metacreation (MuMe) systems, with some impressive results. However, much of the creative work in this field is idiosyncratic, comprising ad hoc standalone systems, and as a result the outcomes can be opaque. In such a diverse environment, it is difficult for artistic researchers to share their ideas or their code, or work out ways that their systems might be incorporated into other's creative workflows. Musebots, responding to these challenges, are small modular units designed to be shared and studied by others. By making collaboration central, and agreeing on communications protocols between computational agents, the musebot project encourages developers to make transparent their system's opera-

tion, while still allowing each musebot to employ different algorithmic strategies.

This paper presents our initial research examining the affordances of a multi-agent decision-making process, developed in a collaboration between four coder-artists. The authors set out to explore strategies by which the musebot ensemble could collectively make decisions about dramatic structure of the music, including planning of more or less major changes, the biggest of which being *when to end*. This is in the context of an initial strategy to work with a distributed decision-making process where each author's musebot agent makes music, and also contributes to the decision-making process. Questions that needed to be addressed, then, included:

- how each musebot should relate its decision-making to its music generation strategy;
- how it should respond to the collective, updating both its future decisions and musical plan;
- what decision messages should be used;
- whether decision-making agents should share common code;
- and whether decision-making agents should strictly conform to given agreements about the decision-making process.

We describe these explorations and their outcomes below, but first provide a brief overview of the musebot research context to date.

A Brief History of Musebot Ensembles

The premiere musebots ensemble occurred in July 2015 as an installation at the International Conference on Computational Creativity (ICCC) in Park City, and was followed in August 2015 at the International Symposium of Electronic Art (ISEA) in Vancouver. Since then, it has been presented at the Generative Art Festival in Venice in December 2015, the New Interfaces for Musical Expression (NIME) conference in Brisbane in July 2016, and the Sound and Music Computing (SMC) conference in Hamburg in August 2016. The first musebot ensembles are more fully de-

scribed elsewhere (Eigenfeldt et al. 2015), along with issues and questions raised by these performances.

The Chill-out Sessions – so named due to an initial desire to provide musebots as an alternative listening space to the dance rhythms of Algoraves (Collins and McLean 2014) – have consisted of ensembles curated by the first author from a growing pool of publicly shared musebots. The musebot test suite¹ currently has a repository of over sixty shared musebots – including source code – by nine different developers.

Ensembles

Curation of ensembles for installation has consisted of combining musebots based upon their musical function. For example, several ensembles consist of one or more of the following: a beat musebot, a bass musebot, a pad musebot, a melody musebot, and a harmony generating musebot. A contrasting ensemble might involve combining several noise musebots, or a grouping of only beat musebots (see Table 1). The diversity of musebots is highlighted in their presentation: if listeners don’t find the current ensemble particularly interesting, they can wait five minutes and be presented with something completely different.

Table 1. Musebot types available online

Type	Number available
Bass Generators	5
Beat Generators	13
Harmony Generators	5
Keys/Pads Generators	6
Melody Generators	19
Noise/Texture Generators	16

Musebot ensembles are launched by a master Conductor, whose function is also to provide a centralised timing source by broadcasting a running clock, as well as serving as a central network hub through which all messages are broadcast (see below) via OSC (Wright 1997). A desire for the musebot project is to be platform agnostic, so musebots are standalone applications (Mac and PC) that do not need to run in the development environment. As a result, launching musebot ensembles on a single computer does take some time, as individual audio applications are started up.

Ensembles are organized as text files, in the following format:

```
tempo      [BPM]      duration      [seconds]
musebot_name      message      value...
musebot_name      message      value...
...
```

Messages currently used in the ensembles include:

- gain (0.0 - 1.0)
- delay (in seconds)
- kill (in seconds).

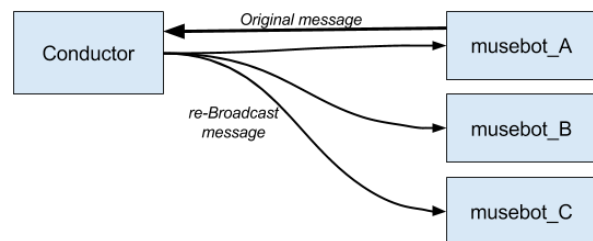
¹ <http://musicalmetacreation.org/musebot-test-suite/>

The **gain** value will be sent to the musebot after all musebots are loaded. This allows for rough mixing of musebot levels. A **delay** value, in seconds, will delay the launching of that musebot by that length of time. Launching begins after the delay, and may take several seconds, so it cannot be assumed that the musebot will begin playing at that specific time. A **kill** value, in seconds, will cause the Conductor to send a kill message to the musebot after that time once the musebot has been launched, taking into account any delay. Thus, “delay 20 kill 20” will cause the musebot to launch with a delay of twenty seconds, then be killed 20 seconds later. Combining delay and kill messages allow for the ensemble to dynamically vary during presentation time.

Broadcasting Messages

As mentioned, the Conductor also serves as a central hub through which messages are passed by the individual musebots. Broadcast messages are general messages sent from one musebot to the ensemble, and include the ID of the originating musebot, the message type, and the data (see Figure 1).

Fig. 1. Musebot configuration for messaging. musebot_A



sends a *notepool* message with its ID, and this message is passed to the entire ensemble.

The Musebot conductor provides synchronisation (timing) and message passing infrastructure, but beyond this musebot designers need to specify protocols of coordination. Our investigations focused on strategies for minimal effective musical coordination with minimal reliance on ‘top-down’ intervention. This is modeled on a conductor-less small ensemble where independent agents (musicians) agree on a few musical constraints (e.g., tempo, key, meter, genre) and then improvise around these.

An example message, given the musebot configuration in Figure 1 might be:

```
broadcast/notepool/musebot_A 60 62 65 67 68
```

The information shared between musebots has tended to be surface detail, such as a current pool of pitches, density of onsets, and volume. For example, the harmony generating musebots have been used to provide a generated chord progression, sent as both pitch-class sets (independent of range) and notepool messages (which indicate specific pitches). Musebots have used this information to constrain their choice of pitches, resulting in congruent harmony. Other musebots (usually the beat generators) have mes-

saged their current onset density; musebots that respond to this message can decide to follow, or oppose, this density, resulting in an audible interaction.

To reiterate, the broadcast messages passed between musebots are not themselves defined in the specification, and have instead been decided upon by the designers of individual musebots; messages can be as low level – for example, realtime timbral analysis has been messaged – or as high level as required. The messages that a musebot sends, and responds to, are contained within a separate human readable text file – `info.txt` – so that other musebot designers can access and use these messages, and ensembles can be more easily curated. These messages are also provided within the online musebot repository².

Musebot ensembles have previously been limited to five minute performances within installation settings, for two reasons. The first is diplomatic: to allow as many combinations of musebots to be presented to listeners as possible, in as wide a variety of styles as possible; the second is more pragmatic: although having virtual performers audibly agree upon certain parameters – for example, density and harmony – may suggest musically successful machine listening, these levels of interaction become mundane surprisingly quickly. The more subtle interactions that occur in human improvisation ensembles, and their development over time, have not yet been successfully modeled within communally developed musebot ensembles. This is not a limitation of the musebot messaging system (see Eigenfeldt 2016); however, designing a communal set of useful messages that satisfy a broad range of musical goals has proven to be more challenging, despite a shared document for this very purpose. This paper reports on one of the first attempts to collectively address these issues.

Challenges

Musebot ensembles have, for the most part, remained a proof of concept. Our next goal is to continue iterating musebots so as to allow further autonomy, on multiple fronts. This might include getting musebots to decide with which other musebots they play well, how to collaboratively determine key, meter and tempo, and methods to organise how and when major events occur, including the most major of all musical events: starting and ending. Of course, there is no *a priori* need for this to occur in a bottom-up self-organised rather than top-down dictatorial way; however, a research question of interest to us is what types of musebot organisations are creatively fruitful and effective, in a metamusical creative context?

Experiments in collaborative Musebots

Our recent experiments involved the four authors each developing a musebot that took on particular ensemble roles—melodic, harmonic, bass, and drums—in support of

² See, for example, <http://musicalmetacreation.org/musebots/beat-generators/>

an ‘experimental prog rock’ performance. Each developer worked in a different software environment (Pure Data, Max, Extempore and Java) and were free to implement any algorithmic processes. Following discussion about possible minimal coordinating parameters, we chose to use *musical density* and *vote to end*. Musebots were designed to vary their generative output based on the density level (from 0.0 - 1.0) and to respond to a majority vote to end the performance. Periodically musebots broadcast

1. their own density level and,
2. a suggested future density target (based on analysis of the ensemble density profile), and
3. a vote to end or not.

As the project developed, we added the communication of harmonic context (current pitch pool) to these initial parameters. Musebot developers could choose to store the history of broadcast data if they felt it enhanced their musebot’s decision-making processes.

We were interested to see whether density was a sufficient level of abstraction for musical coordination and, if density does provide a level of musical success, what could be learned in terms of effective design principles. Also we hoped to see if our musebot interactions, using only the parameters described, could be scaled to include more dimensions.

The resulting musebot source code is available, and recordings of example performances that resulted are available online. Analysis of the operation of the musebots and the resulting musical interactions are reported below.

Approaches to algorithmically addressing musical parameters

The Musebots reported in this paper were developed during a one-week programming sprint toward the end of 2016 where the authors were co-located and could readily communicate about issues as they arose. Despite authors’ co-location, an interesting aspect of this Musebot experimentation was the independence of implementation and agreed parametric coordination. This separation is reminiscent of that maintained in human ensembles where the logic of a musician’s decision making remains opaque to others, and interaction is based on an observation of behaviors. In this section, each developer outlines their approach to the main defining characteristics of the performance: density, deciding to end, and responding to the pitch pool. These detailed descriptions were *not* available to the other designers, nor were they discussed until after the initial performances.

Pd Musebot - Andrew Brown

The Pd Musebot (*PD_MIDI_bot*) generated melodic material based on a random walk pitch contour and rhythms derived from probabilistic beat subdivisions. Note level data were sent in real-time via MIDI to a software synthesizer for playback. The density was largely a factor of rhythmic sparseness achieved by varying the probabilistic likelihood of notes occurring. Rhythmic coherence was

maintained by coordinating between levels of metric emphasis (downbeat and subdivisions) so as to avoid filtering out metrically salient notes. In addition, density influenced note dynamic level resulting in quieter performance in less dense sections.

Pitch selection was limited to the pitch classes provided in the available pitch pool. When a new pitch pool was broadcast, the musebot updated its pitch class set to match.

Density suggestions were made on the basis of managing ‘boredom’ and ‘confusion’. A proxy for these emotional states was the degree of variation in collective density. When the ensemble density remained static for some time (approximately 36 bars), then a density suggestion that deviated from this was made. Conversely, if the density variation over time was wide then a suggestion to maintain the current ensemble density was made.

Ending decisions involved a combination of current extreme levels of ensemble density (very low or very high) and historical trends of density change (upward or downward over several bars). A final ‘catch all’ was to vary the thresholds for these parameters over time to gradually increase the likelihood of end decisions, thus avoiding performances of unacceptably long duration.

Max Musebot - Arne Eigenfeldt

The Max musebot (*SynthBassBot*) assumed the role of a bass synth; as with many musebots, it made performance decisions resulting in representations very similar to MIDI data (i.e. pitch, volume, and duration) and then produced its own audio based upon a variety of pre-existing samples. At the beginning of every performance, the synthesiser settings would be randomised from within a constrained range, including sample folder, amplitude envelope settings, filter envelope settings, and filter settings. Some of these – amplitude sustain, filter cutoff – were varied during the performance based upon the environment. For example, when the agent became “bored” (described shortly), the possibility of producing a phrase-long filter sweep increased; when the density became low, the envelope sustain increased to “fill the space” with longer notes that occurred less frequently.

Pitches were derived from a pitch set provided by a separate musebot that produced a suggested harmonic progression, and were transposed to a low to low-mid pitch range. Because the root of the chord could be inferred from the messaged harmonic progression, certain notes (i.e. the root) were given a greater probability of occurring. With each new pitch set received, a new pitch ordering, which included repeated pitches and octave transpositions, was created, which was maintained for the duration of that particular harmony. New pitch orderings could be generated every phrase beginning with a 33% probability, which reordered the existing pitch collection.

Similarly, new onset patterns could be initiated with the same probability (but not necessarily at the time). These onsets corresponded to metric placements of sixteenth notes (semiquavers) within the measure: the downbeat is onset 0, the second beat is onset 4, as shown in Figure 2 A.

The extrapolation to eighth notes (quavers) can be seen in Figure 2 B; this pattern is reordered by potentially switching primary onsets (0 4 8 12) and secondary (2 6 10 14). For example, the unordered pattern could switch primary onsets 0 and 8, and the second switch onsets 2 and 14, resulting in a reordered pattern of (8 14 4 6 0 10 12 2). The number of onsets performed within a measure is dependent upon the current density; given a density of 0.5, only the first four onsets would be chosen (8 14 4 6), resulting in a final ordered pattern of (4 6 8 14), shown in Figure 2 C.



Fig. 2. Varying onset patterns in *SynthBassBot*

The *SynthBassBot* could become “bored” if it felt there hadn’t been much change in the ensemble’s mean density. Every two phrases (eight measures), it would check if there had been more than a 10% change in cumulative density; if there hadn’t, it would begin a series of tests to decide its boredom state:

1. on each downbeat, scale its internal boredom parameter exponentially over 16 measures (the default setting for these initial run throughs was 0.2). Count the number of measures which have lacked significant change; when this value becomes greater than the scaled boredom number of measures – using the default setting of 0.2 resulted in two measures – proceed to test #2;
2. Generate a random value between 0. and 1. Compare this to the number of measures which have lacked significant change (scaled by 0.1). If the random value is less, the agent becomes bored.

If the agent became bored, it would cast a vote to alter the current density setting, and potentially execute a filter sweep. The new requested density would be derived from its own ongoing activity model, generated at the beginning of the composition.

The bass musebot monitors the ensemble density, keeping track of the trends over the previous four measures, and determines if the density has been rising, falling, or remaining stable. It uses this judgment to decide on whether to vote to end the performance, based upon the following three criteria:

1. the performance has progressed beyond a minimum number of measures (a logarithmically scaled curve between 1 and 120, dependent upon the musebot’s boredom attribute), and
2. the ensemble density over the past four measures is either falling or stable, and
3. the current density is less than what the average density has been so far.

Extempore Musebot - Toby Gifford

The Extempore musebot *pad_bot* (originally called *negotiation_ybot* before its musical function solidified) adopted the role of generating background harmonic pads — long notes with slowly evolving timbre intended to be a musical backdrop to the melodic and rhythmic elements. After initial experiments, an aesthetic decision was made to produce some degree of sonic output continuously in contrast with other bots' more literal interpretation of density.

Density was manifested through rate of triggering — higher density corresponded to faster triggering of pad notes. When a new note was triggered, the currently playing note(s) faded out, though with some overlap. Because of this, higher density also resulted in greater polyphony as more successive notes overlapped.

An internal 'boredom' measure was implemented to discourage extended periods of extreme density (whether high *or* low). The boredom measure determined suggestions of both density and ending. When a boredom threshold was reached, the musebot would suggest a change of density to be as different as possible to the current density. Similarly, after a fixed minimum performance length of 64 measures, the musebot would suggest ending whenever it was bored.

Note selection was determined by the notepool broadcast message, and by the recent history of note pools. Specifically, the pitch selection for each successive triggered pad note cycled through an internal pitch-set. This pitch-set was initially identical to that of the first notepool message. Upon receiving another notepool message, the internal pitch-set is restricted to the intersection of itself with the new notepool. If there are no pitches in common (i.e. the intersection is empty) then the pitch-set is reset to the new notepool.

Java Musebot – Oliver Bown

The Java musebot plays drums. As with the other bots, the system works only with common, simple and well-studied generative processes that produce pleasant and generatively varied drum sequences. The generative strategy is based on an additive, rather than divisive, metrical approach, resulting in irregular meter structures. The density parameter, shared with the other bots, is combined with a syncopation parameter, which is used internally only. Both parameters are represented internally as integers between 0 and 9. A single kit of drum samples is used, ordered as follows: *closed hat, open hat, kick, snare1, half hat, snare2, rim, clap, ride, crash*. The ordering is used in the weighting of stochastic drum sound selection, with earlier drums in the list being chosen with higher probability. The generative algorithm creates a new pattern each 4-bars. It consists of two stages. First, a base-pattern is created. This consists of a sequence of sixteenth-note drum events, where each event may be zero or more individual hits, with each hit consisting of a drum sound and velocity. Then a "tala" structure is generated. This is a series of subpattern lengths, with a start offset for each subpattern. Subpatterns

are played concatenatively. A subpattern is a playback of the base pattern with the given start offset. For example, if the base pattern is [kick,hat,snare,tom,...] and the tala pattern is [[3,1],[3,1],[2,2],...] with the first number indicating the length and the second number indicating the start offset, then the resulting pattern would be [kick, hat, snare, kick, hat, snare, hat, snare...].

The density and syncopation parameters are used to direct the content of these generated patterns in simple ways. Greater density parameters result in more events in the base-pattern, including greater occurrences of snares and cymbals. Above a certain density threshold, a kick drum on the first sixteenth note and a snare drum on the fifth is guaranteed. The density also contributes to shorter subsequences, whilst the syncopation parameter leads to a greater number of odd-length subsequences over even-length ones. A random number generator is used in the exact selection of values, but this random number generator is seeded according to the density and syncopation parameters, meaning that for any given density-syncopation value pair, the exact same pattern is generated each time, making it easier to gain an understanding of the system's behaviour (both as developer and as listener).

The musebot then performs two core actions as part of its negotiation and planning. It selects a new desired syncopation and density state from a lookup table. This lookup table is generated at start time and maps current state pairs to future desired state pairs. The table largely maps states to neighbouring states (gradual changes), but has a small percentage of 'wormholes' that map to different areas of the state space, resulting in sudden changes. Besides these properties the state transition table is randomly generated. The result is an arbitrary-but-not-random behavioural plan, i.e., the transition behaviour is consistent over time even if it derives from an arbitrary source. This has been argued by Cook and Colton (2015) to be a meaningful strategy for generative systems.

The system then broadcasts its density intention. It updates its actual density based on the agreed strategy of taking the average of all intentions. The system simply uses its desired syncopation as its actual syncopation value in the next 4-bar cycle. The system votes to stop when the density is below a threshold of 10%.

A few other minor behavioural details are as follows. The system looks at the size of the forthcoming change and if the change is small can decide not to create a new pattern for the forthcoming 4-bar cycle. If the forthcoming change is very large, it can plan to perform a fill, which involves generating a new pattern just for the last bar of the 4-bar cycle. Lastly, the system periodically updates its density-syncopation lookup table.

Analysis

All four musebots broadcast their current density ‘periodically’. This relative interval was interpreted independently by the different developers, ranging from every second, to the first beat of every measure, to every beat. It was agreed that musebots must vote on a suggested density before the beginning of the third measure in every four bar phrase, although musebots were free to change their own density at any time. Despite musebot suggestions often being quite dramatic — requesting extreme low or high densities (see Figure 4) — the musebots generally progressed towards these extremes, without ever achieving these levels themselves.

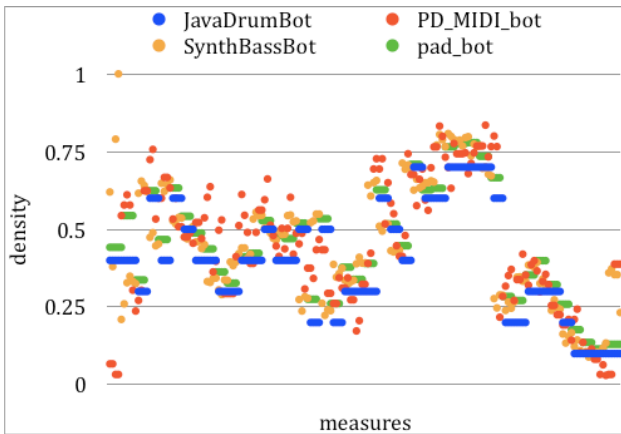


Fig. 3. Actual musebot densities over a 180 measure performance³.

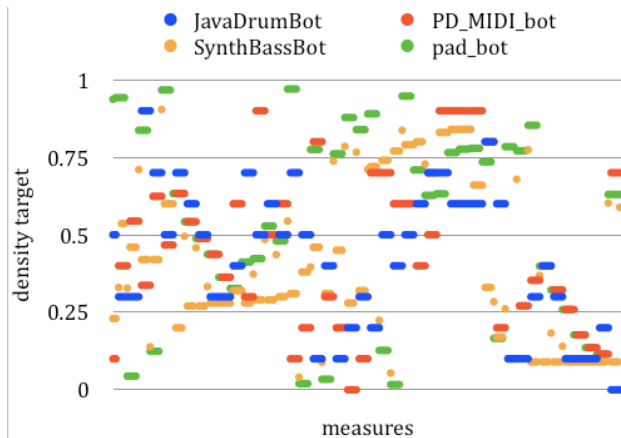


Fig. 4. Suggested density targets.

Figures 3 and 4 display the actual densities (top) versus the suggested density targets (bottom), displaying how the individual musebot interpretation of target suggestions

³ A recording of the performance can be found here: <https://youtu.be/azWnFTMCNic>

results in an undulating mean ensemble density. The suggested targets converge three times – near measures 40, 120, and 170; in each case, these targets follow the apparent direction of the ensemble. The measures in which there is little agreement upon targets – 60-100, and 135-150 – result in the greatest discontinuities in ensemble density: a dramatic leap upward at measure 92, and downward at 135.

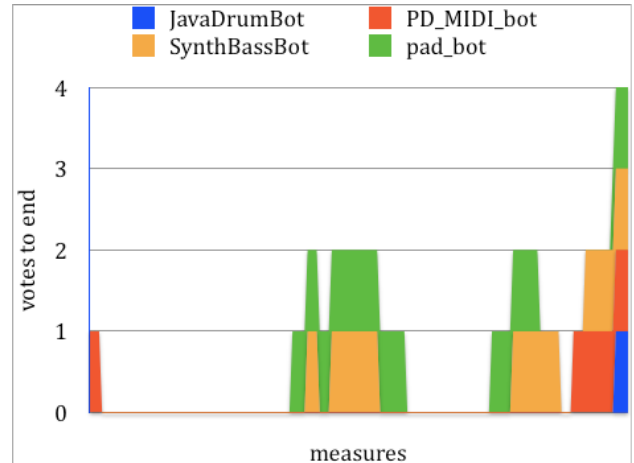


Fig. 5. Votes to end.

Figure 5 displays when musebots voted to end the performance. These votes clustered in three locations that correlate to the lowest actual ensemble densities. Comparing these locations with the suggested density targets of the two musebots that voted to end (*SynthBassBot* and *pad_bot*) shows that these musebots also proposed even lower ensemble densities. This illustrates the intended result of our system design: that the musebots are not simply reactive – voting to end at a predetermined ensemble density – but proactive – continuing their attempt to influence the overall ensemble density.

Comparing Figure 3 and 5 shows how an ending was negotiated and agreed upon once all four musebots voted to end. In fact, only a majority vote, rather than unanimity, is required for an ending; in this case, all four musebots recognised and agreed upon the potential ending. *SynthBassBot* anticipated the ending by requesting densities below the current ensemble mean – which was already decreasing – because it recognised the conditions approaching a possible ending. *SynthBassBot* has its own internal density model – including an ending; this can be seen by the alternation between two states in its density target voting: an initial vote based upon its model, then a second vote averaged between its model and the mean of the ensemble targets.

Comparing the actual ensemble density to the mean target requests (Figure 6) displays the expected correlation. In most cases, the targets precede the ensemble response; for example, the dramatic drop $\frac{3}{4}$ of the way through. Points of disagreement in between targets (e.g. the section prior to the midpoint) do not provoke any significant changes when

compared to when the targets more closely align (e.g. the immediately preceding section).

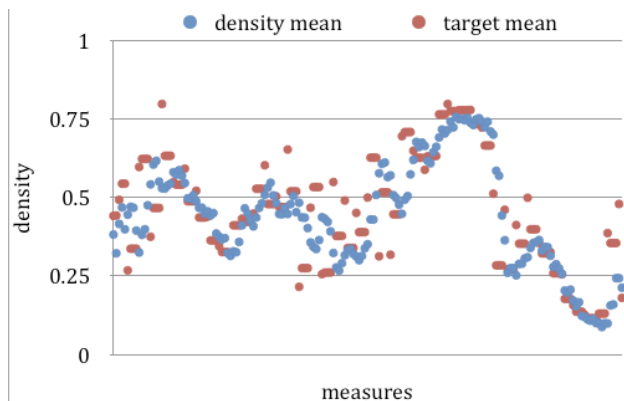


Fig. 6. Mean ensemble density versus mean target suggestions.

Discussion

An iterative practice-led development approach was adopted for these experiments. Correspondingly, it is informative to report on both the practice-based learning that arose and the results of performances generated by the musebot ensemble.

Successes and Failures

During the communal design stage, we settled upon a limited set of parameters for interaction once we determined that sending around too much complex musical data was too challenging to program. This reaffirmed notions of maintaining simplicity when dealing with complex information. We also found that limiting the communication around a single parameter – density – greatly reduced the creative demands of the task (a contrary position was suggested, in that we are not attempting to necessarily simulate a human band, so why rely upon such obvious modes of interaction?). Limiting interaction to 4-bar cycles was also convenient, although it forced the resulting music into a 4-bar mode.

Likewise, the authors found that from a practical working point of view the musebot paradigm of communication at a meta-musical layer was creatively effective, and information flow from the musical surface to the concept layer was not required *for successful interaction*. Because musebots interacting this way communicate *what they are doing*, other musebots are not required to analyse each other's output. For example, musebots need not know the specific pattern a drumbot is playing, but instead the parameters the drumbot is using to generate that pattern – e.g. density and syncopation. At the same time, this adds additional load on the programmers, who must consider both generating actions and also communicating those actions, as well as intentions. Thus the design problem seems to involve a trade off between the value of mediating interac-

tion between the musical surface and the value of communicating agreed meta-parameters. We discussed extending the communication to further parameters – valence, to match the arousal suggested by density – but this was left for future work. An important question is whether further parameters can be included without a debilitating explosion in complexity.

Possible New Designs and Strategies

Our initial questions included:

- how each musebot should relate its decision making to its music generation strategy;
- how it should respond to the collective, updating both its future decisions and musical plan;
- what decision messages should be used;
- whether decision making agents should share common code;
- whether decision making agents should strictly conform to given agreements about the decision making process.

The first three questions are partially addressed in the above discussion. We see value in this formulation and believe it can form the basis for effective music creation as well as research around musical decision-making. However, the process also pointed to alternative designs that could be more effective.

To begin with, developing four different systems that implemented the same basic framework of decision making, let alone performing any machine listening, style modeling and so on, suggested that a shared codebase or service architecture would be effective. A service architecture would be a way to compile successful strategies in machine listening, style modeling, decision making and algorithmic composition strategies into a common repository. Individual agents would be able to query services for information such as what is going on in the music at the moment, or what would make a good note or chord to follow a given sequence given a certain style database or even information contained in the current performance.

It could also provide services to support negotiation. In essence, under the current scheme, each programmer is building a generative music system as well as very simple virtual psychology underlying the decisions surrounding density planning; for example, the agent can be fickle, or conformant. This is hard, repetitive, and prone to errors or mistaken understanding. There is great potential for farming out such decision-making to well-developed models that incorporate aspects of psychology.

Thus the experiment points to an agent-based architecture which would break quite significantly from one in which each agent is the equivalent of a human musician, listening, making decisions and generating music. An ensemble would still consist of a series of agents that would have the same agency to perform musical acts, but much of their cognitive machinery would exist in 3rd party agents that provided services and might also act as a form of distributed cognition. This would allow certain aspects of top-

down organisation and would form a heterogeneous distributed system.

Future Directions

While the performances produced by these musebots demonstrate interesting variations in the musical surface in density, these are minor developments in musebot ensemble design, albeit with the additional of a negotiated ending. From a musical perspective, a continued limitation a lack of large-scale change within the way the ensemble performs over time. A common criticism of young improvisers and composers is that the final minute, for example, does not vary a great deal from the first minute. In our case, the musebots did not alter the way in which they interpreted density, or even the way in which they fulfilled their roles within the ensemble: *PD_MIDI_bot* freely improvised melodically in the same general fashion, *pad_bot* played pads, *SynthBassBot* played the same general bass line using the same timbres, and *JavaDrumBot* performed the same basic rhythmic patterns. At no time, for example, did any part actually stop playing, or take over the foreground role. Such foreground/background negotiation is standard in improvised music, and would be the next step in achieving more musical interaction within this musebot ensemble.

Another utility of algorithmic experimentation of this kind is that it requires the articulation and testing of theories of behavior. This is a musicological activity when implementing musical performance outcomes, as in the musebots case, and is often contributive to this field (Brown et al. 2009). But the method can be applied more generally to behavioural understanding in many fields, and particularly to the dynamics of creative collaboration and interaction.

Conclusion

Musebots are based upon a state-driven communication system, rather than an output-driven system that requires feature analysis found in many metacreative systems: for example, the *P* in Blackwell and Young's *PfQ* model (Blackwell and Young 2005) or the Listener in Rowe's model (1993). Our goal in this paper was not to compare the two models, but to accept the musebot ideal as outlined in the original manifesto (2015), and explore its potential as a platform for better scaffolding musical intelligence in large modular systems.

By limiting the messages passed by musebots to current density, proposed density, and a vote to end, we feel that our ensemble of four independent systems – coded without explicit collaboration in algorithm design – demonstrated some ways in which multiple musical metacreative creators can make ensemble performances mediated by machine interaction.

The code for the musebots described in this paper is available here:

<https://vault.sfu.ca/index.php/s/O4AY9DBSR5wBITI>

A recording of the performance can be found here:

<https://youtu.be/azWnFTMCNlc>

Acknowledgements

The authors wish to acknowledge the Social Sciences and Humanities Research Council of Canada (SSHRC) for funding this research.

References

- Blackwell, T., and Young, M. 2005. Live Algorithms. Available online at www.timblackwell.com.
- Bown, O., Carey, B., and Eigenfeldt, A. 2015. Manifesto for a Muse-bot Ensemble: A platform for live interactive performance between multiple autonomous musical agents. *Proceedings of the International Symposium of Electronic Art*, Vancouver.
- Brown, A. R., Gifford, T., Narmour, E. and Davidson, R. 2009. Generation in Context: an exploratory method for musical enquiry. *Proceedings of the 2nd International Conference on Music Communication Science*, Sydney.
- Collins, N, and McLean, A. 2014. Algorave: Live performance of algorithmic electronic dance music. *Proceedings of the International Conference on New Interfaces for Musical Expression*, London.
- Cook, M., and Colton, S. 2015. Generating code for expressing simple preferences: Moving on from hardcoding and randomness. *Proceedings of the Sixth International Conference on Computational Creativity*, Park City.
- Eigenfeldt, A., Bown, O., and Carey, B. 2015. Collaborative Composition with Creative Systems: Reflections on the First Musebot Ensemble. *Proceedings of the ICCA*, Park City.
- Eigenfeldt, A. 2016. Exploring Moment-Form in Generative Music. *Proceedings of the Sound and Music Computing Conference*, Hamburg.
- Rowe, R. 1993. *Interactive Music Systems*. Cambridge, Massachusetts: MIT Press.
- Wright, M. 1997. Open Sound Control - A New Protocol for Communicating with Sound Synthesizers. *Proceedings of the International Computer Music Conference*, Thessaloniki.

CAN: Creative Adversarial Networks

Generating “Art” by Learning About Styles and Deviating from Style Norms

Ahmed Elgammal¹ Bingchen Liu¹ Mohamed Elhoseiny² Marian Mazzone³

¹ Department of Computer Science, Rutgers University, NJ, USA

² Facebook AI Research, CA, USA

³ Department of Art History, College of Charleston, SC, USA

Abstract

We propose a new system for generating art. The system generates art by looking at art and learning about style; and becomes creative by increasing the arousal potential of the generated art by deviating from the learned styles. We build over Generative Adversarial Networks (GAN), which have shown the ability to learn to generate novel images simulating a given distribution. We argue that such networks are limited in their ability to generate creative products in their original design. We propose modifications to its objective to make it capable of generating creative art by maximizing deviation from established styles and minimizing deviation from art distribution. We conducted experiments to compare the response of human subjects to the generated art with their response to art created by artists. The results show that human subjects could not distinguish art generated by the proposed system from art generated by contemporary artists and shown in top art fairs.

Introduction

Since the dawn of Artificial Intelligence, scientists have been exploring the machine’s ability to generate human-level creative products such as poetry, stories, jokes, music, paintings, etc., as well as creative problem solving. This ability is fundamental to show that Artificial Intelligence algorithms are in fact intelligent. In terms of visual art, several systems have been proposed to automatically create art, not only in the domain of AI and computational creativity (e.g. (Baker and Seltzer(1993); DiPaola and Gabora(2009); Colton et al.(2015)Colton, Halskov, Ventura, Gouldstone, Cook, and Perez-Ferrer; Heath and Ventura(2016)), but also in computer graphics (Sims(1991)), and machine learning, (e.g. (Mordvintsev et al.(2015)Mordvintsev, Olah, and Tyka; Johnson et al.(2016)Johnson, Alahi, and Fei-Fei)).

Within the computational creativity literature, different algorithms have been proposed focused on investigating various and effective ways of exploring the creative space. Several approaches have used an evolutionary process wherein the algorithm iterates by generating candidates, evaluating them using a fitness function, and then modifying them to improve the fitness score for the next iteration (e.g. (Machado et al.(2015)Machado, Romero, and Manaris; DiPaola and Gabora(2009))). Typically, this process is done

within a genetic algorithm framework. As pointed out by DiPaola and Gabora 2009, the challenge for any algorithm centers on “how to write a logical fitness function that has an aesthetic sense”. Some earlier systems utilized a human in the loop with the role of guiding the process (e.g. (Baker and Seltzer(1993); Graf and Banzhaf(1995))). In these interactive systems, the computer explores the creative space and the human plays the role of the observer whose feedback is essential in driving the process. Recent systems have emphasized the role of perception and cognition in the creative process (Colton(2008); Colton et al.(2015)Colton, Halskov, Ventura, Gouldstone, Cook, and Perez-Ferrer; Heath and Ventura(2016)).

The goal of this paper is to investigate a computational creative system for art generation without involving a human artist in the creative process, but nevertheless involving human creative products in the learning process. An essential component in art-generating algorithms is relating their creative process to art that has been produced by human artists throughout time. We believe this is important because a human’s creative process utilizes prior experience of and exposure to art. A human artist is continuously exposed to other artists’ work, and has been exposed to a wide variety of art for all of his/her life. What remains largely unknown is how human artists integrate their knowledge of past art with their ability to generate new forms. A theory is needed to model how to integrate exposure to art with the creation of art.

Colin Martindale (1943-2008) proposed a psychology-based theory that explains new art creation(Martindale(1990)). He hypothesized that at any point in time, creative artists try to increase the arousal potential of their art to push against habituation. However, this increase has to be minimal to avoid negative reaction by the observers (principle of least effort). Martindale also hypothesized that style breaks happen as a way of increasing the arousal potential of art when artists exert other means within the roles of style. The approach proposed in this paper is inspired by Martindale’s principle of least effort and his explanation of style breaks. Among theories that try to explain progress in art, we find Martindale’s theory to be computationally feasible.

Deep neural networks have recently played a transformative role in advancing artificial intelligence across various application domains. In particular, several generative deep networks have been proposed that have the abil-

ity to generate novel images to emulate a given training distribution Generative Adversarial Networks (GAN) have been quite successful in achieving this goal (Goodfellow et al.(2014)Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio). We argue that such networks are limited in their ability to generate creative products in their original design. Inspired by Martindale's theory, in this paper we propose modifications to GAN's objective to make it able to generate creative art by maximizing deviation from established styles while minimizing deviation from art distribution.

Methodology

Background

The proposed approach is motivated from the theory suggested by D. E. Berlyne (1924-1976). Berlyne argued that the psychophysical concept of "arousal" has a great relevance for studying aesthetic phenomena (Berlyne(1971)). "Level of arousal" measures how alert or excited a human being is. The level of arousal varies from the lowest level, when a person is asleep or relaxed, to the highest level when s/he is violent, in a fury, or in a passionate situation (Berlyne(1967)). Among different mechanisms of arousal, of particular importance and relevance to art are properties of external stimulus patterns (Berlyne(1971)).

The term "arousal potential" refers to the properties of stimulus patterns that lead to raising arousal. Besides other psychophysical and ecological properties of stimulus patterns, Berlyne emphasized that the most significant arousal-raising properties for aesthetics are *novelty*, *surprisingness*, *complexity*, *ambiguity*, and *puzzlingness*. He coined the term *collative variables* to refer to these properties collectively.

Novelty refers to the degree a stimulus differs from what an observer has seen/experienced before. Surprisingness refers to the degree a stimulus disagrees with expectation. Surprisingness is not necessarily correlated with novelty, for example it can stem from lack of novelty. Unlike novelty and surprisingness which rely on inter-stimulus comparisons of similarity and differences, complexity is an intra-stimulus property that increases as the number of independent elements in a stimulus grows. Ambiguity refers to the conflict between the semantic and syntactic information in a stimulus. Puzzlingness refers to the ambiguity due to multiple, potentially inconsistent, meanings.

Several studies have shown that people prefer stimulus with a moderate arousal potential (Berlyne(1967); Schneirla(1959)). Too little arousal potential is considered boring, and too much activates the aversion system, which results in negative response. This behavior is explained by the Wundt curve that correlates the arousal potential with the hedonic response (Berlyne(1971); Wundt(1874)). Berlyne also studied arousal moderating mechanisms. Of particular importance in art is habituation, which refers to decreased arousal in response to repetitions of a stimulus (Berlyne(1971)).

Martindale emphasized the importance of habituation in deriving the art-producing system (Martindale(1990)). If artists keep producing similar works of arts, this directly re-

duces the arousal potential and hence the desirability of that art. Therefore, at any point of time, the art-producing system will try to increase the arousal potential of produced art. In other words, habituation forms a constant pressure to change art. However, this increase has to be within the minimum amount necessary to compensate for habituation without falling into the negative hedonic range, according to Wundt curve findings ("stimuli that are slightly rather than vastly supernormal are preferred"). Martindale called this the principle of "least effort". Therefore, there is an opposite pressure that leads to a graduated pace of change in art.

Art Generating Agent

We propose a model for an art-generating agent, and then propose a functioning model using a variant of GAN to make it creative. The agent's goal is to generate art with increased levels of arousal potential in a constrained way without activating the aversion system and falling into the negative hedonic range. In other words, the agent tries to generate art that is novel, but not too novel. This criterion is common in many computationally creative systems, however it is not easy to find a way to achieve that goal given the infinite possibilities in the creative space.

In our model the art-generating agent has a memory that encodes the art it has been exposed to, and can be continuously updated with the addition of new art. The agent uses this encoded memory in an indirect way while generating new art with a restrained increase in arousal potential. While there are several ways to increase the arousal potential, in this paper we focus on building an agent that tries to increase the *stylistic ambiguity* and deviations from style norms, while at the same time, avoiding moving too far away from what is accepted as art. The agent tries to explore the creative space by deviating from the established style norms and thereby generates new art.

There are two types of ambiguities that are expected in the generated art by the proposed network; one is by design and the other one is inherent. Almost all computer-generated art might be ambiguous because the art generated typically does not have clear figures or an interpretable subject matter. Because of this, Heath et al argued that the creative machine would need to have perceptual ability (be able to see) in order to be able to generate plausible creative art (Heath and Ventura(2016)). This limited perceptual ability is what causes the inherent ambiguity. Typically, this type of ambiguity results in users being able to tell right away that the work is generated by a machine rather than a human artist. Even though several styles of art developed in the 20th century might lack recognizable figures or lucid subject matter, human observers usually are not fooled into confusing computer-generated art with human generated art.

Because of this inherent ambiguity people always think of computer-generated art as being hallucination-like. The Guardian commented on a the images generated by Google DeepDream (Mordvintsev et al.(2015)Mordvintsev, Olah, and Tyka) by "Most, however, look like dorm-room mandalas, or the kind of digital psychedelia you might expect

to find on the cover of a Terrence McKenna book”¹. Others commented on it as being “dazzling, druggy, and creepy”². This negative reaction might be explained as a result of too much arousal, which results in negative hedonic according to the Wundt curve.

The other type of ambiguity in the art generated by the proposed agent is stylistic ambiguity, which is intentional by design. The rationale is that creative artists would eventually break from established styles and explore new ways of expression to increase the arousal potential of their art, as Martindale suggested. As suggested by DiPaola and Gabora, “creators often work within a very structured domain, following rules that they eventually break free of” (DiPaola and Gabora(2009)).

The proposed art-generating agent is realized by a model called Creative Adversarial Network (CAN), which we will describe next. The network is designed to generate art that does not follow established art movements or styles, but instead tries to generate art that maximally confuses human viewers as to which style it belongs to.

GAN: Emulative and not Creative

Generative Adversarial Network (GAN) has two sub networks, a generator and a discriminator. The discriminator has access to a set of image (training images). The discriminator tries to discriminate between “real” images (from the training set) and “fake” images generated by the generator. The generator tries to generate images similar to the training set without seeing these images. The generator starts by generating random images and receives a signal from the discriminator whether the discriminator finds them real or fake. At equilibrium the discriminator should not be able to tell the difference between the images generated by the generator and the actual images in the training set, hence the generator succeeds in generating images that come from the same distribution as the training set.

Let us now assume that we trained a GAN model on images of paintings. Since the generator is trained to generate images that fool the discriminator to believe it is coming from the training distribution, ultimately the generator will just generate images that look like already existing art. There is no motivation to generate anything creative. There is no force that pushes the generator to explore the creative space. Let us think about a generator that can cheat and already has access to samples from the training data. In that case the discriminator will right away be fooled into believing that the generator is generating art, while in fact it is already existing art, and hence not novel and not creative.

There have been extensions to GANs that facilitate generating images conditioned on categories (e.g., (Radford et al.(2016)Radford, Metz, and Chintala)) or captions (e.g., (Reed et al.(2016)Reed, Akata, Yan, Logeswaran, Schiele, and Lee)). We can think of a GAN that can be designed and trained to generate images of different art styles or different art genres by providing such labels with training.

¹Alex Rayner, the Guardian, March 28, 2016

²David Auerbach, Slate, July 23, 2015

This might be able to generate art that looks like, for example, Renaissance, Impressionism, or Cubism. However that does not lead to anything creative either. No creative artist will create art today that tries to emulate the Baroque or Impressionist style, or any traditional style, unless doing so ironically. According to Berlyne and Martindale, artists would try to increase the arousal potential of their art by creating novel, surprising, ambiguous, and/or puzzling art. This highlights the fundamental limitation of using GANs in generating creative works.

From being Emulative to being Creative

In the proposed Creative Adversarial Network CAN?, the generator is designed to receive two signals from the discriminator that act as two contradictory forces to achieve three points: 1) generate novel works, 2) the novel work should not too novel, i.e., it should not be too far away from the distribution or it will generate too much arousal, thereby activating the aversion system and falling into the negative hedonic range according to the Wundt curve, 3) the generated work should increase the stylistic ambiguity.

Similar to Generative Adversarial Networks (GAN), the proposed network has two adversary networks, a discriminator and a generator. The discriminator has access to a large set of art associated with style labels (Renaissance, Baroque, Impressionism, Expressionism, etc.) and uses it to learn to discriminate between styles. The generator does not have access to any art. It generates art starting from a random input, but unlike GAN, it receives two signals from the discriminator for any work it generates. The first signal is the discriminator’s classification of “art or not art”. In traditional GAN, this signal enables the generator to change its weights to generate images that more frequently will deceive the discriminator as to whether it is coming from the same distribution. Since the discriminator in our case is trained on art, this will signal whether the discriminator thinks the generated art is coming from the same distribution as the actual art it knows about. In that sense, this signal flags whether the discriminator thinks the image presented to it is “art or not art”. Since the generator only receives this signal, it will eventually converge to generate images that will emulate art.

The second signal the generator receives is a signal about how well the discriminator can classify the generated art into established styles. If the generator generates images that the discriminator thinks are art and also can easily classify into one of the established styles, then the generator would have fooled the discriminator into believing it generated actual art that fits within established styles. In contrast, the creative generator will try to generate art that confuses the discriminator. On one hand it tries to fool the discriminator to think it is “art,” and on the other hand it tries to confuse the discriminator about the style of the work generated.

These two signals are contradictory forces, because the first signal pushes the generator to generate works that the discriminator accepts as “art,” however if it succeeds within the rules of established styles, the discriminator will also be able to classify its style. Then the second signal will heftily penalize the generator for doing that. This is because the second signal pushes the generator to generate style-

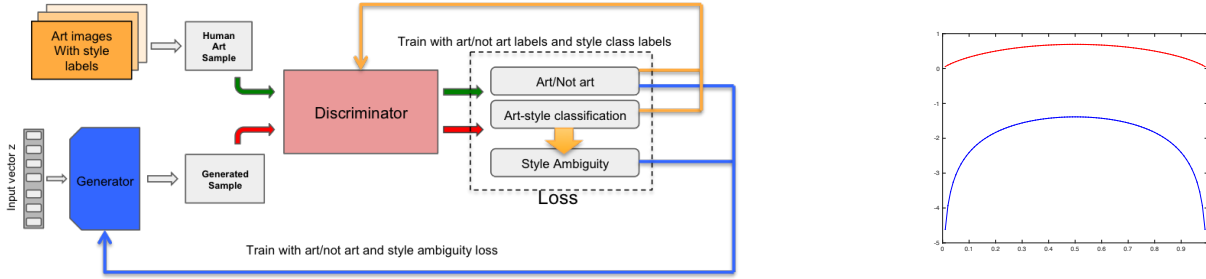


Figure 1: Left: Block diagram of the CAN system. Right: Style ambiguity cross entropy loss function (two class case). Red curve: entropy. Blue curve: cross entropy with a uniform distribution (inverted). Both functions are maximized when the classes are equiprobable. In contrast to entropy which goes to zero at the boundaries, the cross entropy goes to negative infinity at the boundary, which causes hefty penalty for samples classified correctly.

ambiguous works. Therefore, these two signals together should push the generator to explore parts of the creative space that lay close to the distribution of art (to maximize the first objective), and at the same time maximizes the ambiguity of the generated art with respect to how it fits in the realm of standard art styles.

Technical Details

Generative Adversarial Networks

Generative Adversarial Network (GAN) (Goodfellow et al.(2014)Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio) is one of the most successful image synthesis models in the past few years. GAN is typically trained by setting a game between two players. The first player, called the generator, G , generates samples that are intended to come from the same probability distribution as the training data (i.e. p_{data}), without having access to such data. The other player, denoted as the discriminator, D , examines the samples to determine whether they are coming from p_{data} (real) or not (fake). Both the discriminator and the generator are typically modeled as deep neural networks. The training procedure is similar to a two-player min-max game with the following objective function

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))], \quad (1)$$

where z is a noise vector sampled from distribution p_z (e.g., uniform or Gaussian distribution) and x is a real image from the data distribution p_{data} . In practice, the discriminator and the generator are alternatively optimized for every batch. The discriminator aims at maximizing Eq 1 by minimizing $-\mathbb{E}_{x \sim p_{data}} [\log D(x)] - \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))]$, which improves the utility of the D as a fake vs. real image detector. Meanwhile, the generator aims at minimizing Eq 1 by maximizing $\log(D(G(z)))$, which works better than $-\log(1 - D(G(z)))$ since it provides stronger gradients. By optimizing D and G alternatively, GAN is trained to generate images that emulate the training distribution.

Creative Adversarial Networks

We modified the GAN loss function to achieve the vision explained in the previous section. Figure 1 illustrates the ar-

chitecture. We added a style classification loss and a style ambiguity loss. Maximizing the stylistic ambiguity can be achieved by maximizing the style class posterior entropy. Hence, we need to design the loss such that the generator G produces an image $x \sim p_{data}$ and, meanwhile, maximizes the entropy of $p(c|x)$ (i.e. style class posterior) for the generated images. The direct way to increase the stylistic ambiguity is to maximize the class posterior entropy. However, instead of maximizing the class posterior entropy, we minimize the cross entropy between the class posterior and a uniform target distribution. Similar to entropy that is maximized when the class posteriors (i.e., $p(c|G(z))$) are equiprobable, cross entropy with uniform target distribution will be minimized when the classes are equiprobable. So both objectives will be optimal when the classes are equiprobable. However, the difference is that the cross entropy will go up sharply at the boundary since it goes to infinity if any class posterior approaches 1 (or zero), while entropy goes to zero at this boundary condition (see Figure 1-right). Therefore, using the cross entropy results in a hefty penalty if the generated image is classified to one of the classes with high probability. This in turn would generate very large loss, and hence large gradients if the generated images start to be classified to any of the style classes with high confidence. Hence, we can redefine the cost function with a different adversarial objective as

$$\min_G \max_D V(D, G) = \mathbb{E}_{x, \hat{c} \sim p_{data}} [\log D_r(x) + \log D_c(c = \hat{c}|x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D_r(G(z))) - \sum_{k=1}^K (\frac{1}{K} \log(D_c(c_k|G(z))) + (1 - \frac{1}{K}) \log(1 - D_c(c_k|G(z))))], \quad (2)$$

where z is a noise vector sampled from distribution p_z (e.g., uniform or Gaussian distribution) and x and \hat{c} are a real image and its corresponding style label from the data distribution p_{data} . $D_r(\cdot)$ is the transformation function that tries to discriminate between real art and generated images. $D_c(\cdot)$ is the the function that discriminates between different style categories and estimates the style class posteriors (i.e., $D_c(c_k|\cdot) = p(c_k|\cdot)$).

Discriminator Training: In Eq 2, the discriminator D encourages maximizing Eq 2 by minimizing $-\mathbb{E}_{x \sim p_{data}} [\log D_r(x) + \log D_c(c = \hat{c}|x)]$ for the real images and $-\mathbb{E}_{z \sim p_z} [\log(1 - D_r(G(z)))]$ for the generated images. The discriminator is trained to, not only discriminate the real art samples from the generated (fake) ones, but also to identify their style class through the K-way loss (where K is the number of style classes). Therefore, the discriminator is simultaneously learning about both the art distribution and art styles.

Generator Training: The generator G encourages minimizing Eq 2 by maximizing $\log(1 - D_r(G(z))) - \sum_{k=1}^K (\frac{1}{K} \log(D_c(c_k|G(z))) + (1 - \frac{1}{K}) \log(1 - D_c(c_k|G(z))))$. This pushes the generated images to look as real art (first term) and meanwhile to have a large cross entropy for $p(c|G(z))$ with a uniform distribution to maximize style ambiguity (second term). Note that the CAN generator does not require any class labels, similar to unconditional generative model.

Model Architecture: Algorithm 1 illustrates CAN training process. The Generator G and similar to architecture (Radford et al.(2016)Radford, Metz, and Chintala), first $z \in \mathbb{R}^{100}$ normally sampled from 0 to 1 is up-sampled to a $4 \times$ spatial extent convolutional representation with 2048 feature maps resulting in a $4 \times 4 \times 2048$ tensor. Then a series of four fractionally-stride convolutions (in some papers, wrongly called deconvolutions). Finally, convert this high level representation into a 256×256 pixel image. In other words, starting from $z \in \mathbb{R}^{100} \rightarrow 4 \times 4 \times 1024 \rightarrow 8 \times 8 \times 1024 \rightarrow 16 \times 16 \times 512 \rightarrow 32 \times 32 \times 256 \rightarrow 64 \times 64 \times 128 \rightarrow 128 \times 128 \times 64 \rightarrow 256 \times 256 \times 3$ (the generated image size). As described earlier, the discriminator has two losses (real/fake loss and multi-label loss). The discriminator in our work starts by a common body of convolution layers followed by two heads (one for the real/fake loss and one for the multi-label loss). *The common body* of convolution layers is composed of a series of six convolution layers (all with stride 2 and 1 pixel padding). conv1 ($32 \times 4 \times 4$ filters), conv2 ($64 \times 4 \times 4$ filters), conv3 ($128 \times 4 \times 4$ filters), conv4 ($256 \times 4 \times 4$ filters), conv5 ($512 \times 4 \times 4$ filters), conv6 ($512 \times 4 \times 4$ filters). Each convolutional layer is followed by a leaky rectified activation (LeakyReLU) (Maas et al.(2013)Maas, Hannun, and Ng; Xu et al.(2015)Xu, Wang, Chen, and Li) in all the layers of the discriminator. After passing a image to the common conv D body, it will produce a feature map or size ($4 \times 4 \times 512$). *The real/fake D_r head* collapses the ($4 \times 4 \times 512$) by a fully connected to produce $D_r(c|x)$ (probability of image coming for the real image distribution). *The multi-label probabilities $D_c(c_k|x)$ head* is produced by passing the($4 \times 4 \times 512$) into 3 fully collected layers sizes 1024, 512, K , respectively, where K is the number of style classes.

Initialization and Training: The weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. We used a mini-batch size of 128 and used mini-batch stochastic gradient descent (SGD) for training with 0.0001 as learning rate. In the LeakyReLU, the slope of the leak was set to 0.2 in all models. While previous GAN work has used momentum to accelerate training, we used the

Algorithm 1 CAN training algorithm with step size α , using mini-batch SGD for simplicity.

- 1: **Input:** mini-batch images x , matching label \hat{c} , number of training batch steps S
- 2: **for** $n = 1$ **to** S **do**
- 3: $z \sim \mathcal{N}(0, 1)^Z$ {Draw sample of random noise}
- 4: $\hat{x} \leftarrow G(z)$ {Forward through generator}
- 5: $s_D^r \leftarrow D_r(x)$ {real image, real/fake loss }
- 6: $s_D^c \leftarrow D_c(\hat{c}|x)$ {real image, multi class loss }
- 7: $s_G^f \leftarrow D_r(\hat{x})$ {fake image, real/fake loss }
- 8: $s_G^c \leftarrow \sum_{k=1}^K \frac{1}{K} \log(p(c_k|\hat{x})) + (1 - \frac{1}{K})(\log(p(c_k|\hat{x})))$ {fake image Entropy loss }
- 9: $\mathcal{L}_D \leftarrow \log(s_D^r) + \log(s_D^c) + \log(1 - s_G^f)$
- 10: $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$ {Update discriminator }
- 11: $\mathcal{L}_G \leftarrow \log(s_G^f) - s_G^c$
- 12: $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$ {Update generator }
- 13: **end for**

Table 1: Artistic Styles Used in Training

Style name	Image number	Style name	Image number
Abstract-Expressionism	2782	Mannerism-Late-Renaissance	1279
Action-Painting	98	Minimalism	1337
Analytical-Cubism	110	Naive Art-Primitivism	2405
Art-Nouveau-Modern	4334	New-Realism	314
Baroque	4241	Northern-Renaissance	2552
Color-Field-Painting	1615	Pointillism	513
Contemporary-Realism	481	Pop-Art	1483
Cubism	2236	Post-Impressionism	6452
Early-Renaissance	1391	Realism	10733
Expressionism	6736	Rococo	2089
Fauvism	934	Romanticism	7019
High-Renaissance	1343	Synthetic-Cubism	216
Impressionism	13060	Total	75753

Adam optimizer and trained the model for 100 epochs (100 passes over the training data). To stabilize the training, we used Batch Normalization (Ioffe and Szegedy(2015)) that normalizing the input to each unit to have zero mean and unit variance. We performed data augmentation by adding 5 crops within for each image (bottom-left, bottom-right, mid, top-left, top-right) on our image dataset. The width and height of each crop is 90% of the width and the height of the original painting.

Results and Validation

Training the model

We trained the networks using paintings from the publicly available WikiArt dataset³. This collection (as downloaded in 2015) has images of 81,449 paintings from 1,119 artists ranging from the Fifteenth century to Twentieth century. Table 1 shows the number of images used in training the model from each style.

Validation

Assessing the creativity of artifacts generated by the machine is an open and hard question. As noted by Colton 2008, aesthetic assessment of an artifact is different from the creativity assessment (Colton(2008)).

³<https://www.wikiart.org/>

Table 2: Means and standard deviations of responses of Experiment I

Painting set	Q1 (std)	Q2 (std)
CAN	53% (1.8)	3.2 (1.5)
GAN (Radford et al.(2016)Radford, Metz, and Chintala)	35% (1.5)	2.8 (0.54)
Abstract Expressionist	85% (1.6)	3.3 (0.43)
Art Basel 2016	41% (2.9)	2.8 (0.68)
Artist sets combined	62% (3.2)	3.1 (0.63)

Table 3: Means and standard deviations of the responses of Experiment II

Painting set	Q1 (std)	Q2 (std)	Q3 (std)	Q4 (std)
CAN	3.3 (0.47)	3.2 (0.47)	2.7 (0.46)	2.5 (0.41)
Abstract Expressionist	2.8 (0.43)	2.6 (0.35)	2.4 (0.41)	2.3 (0.27)
Art Basel 2016	2.5 (0.72)	2.4 (0.64)	2.1 (0.59)	1.9(0.54)
Artist sets combined	2.7 (0.6)	2.5 (0.52)	2.2 (0.54)	2.1 (0.45)

We conducted human subject experiments to evaluate aspects of the creativity of the proposed model. The goal of this experiments is to test whether human subjects would be able to distinguish whether the art is generated by a human artist or by a computer system, as well as to rate aspects of that art. However, the hard question is which art by human artists we should use for this comparison. Since the goal of this study is to evaluate the creativity of the artifacts produced by the proposed system, we need to compare human response to such artifacts with art that is considered to be novel and creative at this point in time. If we compare the produced artifacts to, for example, Impressionist art, we would be testing the ability of the system to emulate such art, and not the creativity of the system. Therefore we collected two sets of works by real artists ,as well as two machine-generated sets as follows:

1. Abstract Expressionist Set: A collection of 25 painting by Abstract Expressionist masters made between 1945-2007, many of them by famous artists. This set was previously used in recent studies to compare human and machine’s ability to distinguish between abstract art by created artists, children or animals (Snapper et al.(2015)Snapper, Oranç, Hawley-Dolan, Nissel, and Winner; Shamir et al.(2016)Shamir, Nissel, and Winner). We use this set as a baseline set. Human subjects are expected to easily determine that these are created by artists.
2. Art Basel 2016 Set: This set consists of 25 paintings of various artists that were shown in Art Basel 2016, which is the flagship art fair for contemporary art world wide. Being shown in Art Basel 2016 is an indication that these are art works at the frontiers of human creativity in paintings, at least as judged by the art experts and the art market.
3. DC GAN Set: a set of 100 images generated by the state-of-the-art Deep Convolution GAN (DCGAN) architecture (Radford et al.(2016)Radford, Metz, and Chintala)
4. CAN Set: a set of 125 images generated by the model.

We used the same set of training images for both the GAN and CAN models and we conducted two human subject experiments as follows.

Experiment I:

The goal of this experiment is to test the ability of the system to generate art that human users would not distinguish from top creative art that is being generated by artists today.

In this experiment each subject is shown one image at time selected from the four sets of images described above and asked:

Q1: Do you think the work is created by an artist or generated by a computer? The user has to choose one of two answers: artist or computer.

Q2: The user asked to rate how they like the image in a scale 1 (extremely dislike) to 5 (extremely like).

Results: 18 MTurk users participated in this experiment. The results are summarized in Table 2. There are several conclusions we can draw from these results: 1) As expected, subjects rated the Abstract Expressionist set higher as being created by an artist (85%). 2) The proposed CAN model significantly out-perform GAN model in generating images that human think are generated by artist (53% vs. 35%). Of course we cannot obviously conclude from this results that CAN is more creative than GAN. A perfect system that would perfectly copy human art, without being innovative, would score higher in that question. However, we can exclude this possibility since the generated images by both CAN and GAN are not by any means copying human art. 3) More interestingly, human subject rated the images generated by CAN higher as being created by a human than the ones from the Art Basel set (53% vs. 41%) when combining the two sets of art created by artists, the images generated by CAN scored only 9% less (53% vs. 62%).

Experiment II:

This experiment is similar to an experiment conducted by (Snapper et al.(2015)Snapper, Oranç, Hawley-Dolan, Nissel, and Winner) to determine to what degree human subject find the works of art to be intentional, having visual structure, communicative, and inspirational.

Q1: As I interact with this painting, I start to see the artist’s intentionality: it looks like it was composed very intentionally.

Q2: As I interact with this painting, I start to see a structure emerging.

Q3: Communication: As I interact with this painting, I feel that it is communicating with me.

Q4: Inspiration: As I interact with this painting, I feel inspired and elevated.

For each of the question the users answered in a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). The users were asked to look at each image at least 5 second before answering. 21 users participated in this experiment.

(Snapper et al.(2015)Snapper, Oranç, Hawley-Dolan, Nissel, and Winner), hypothesized that subjects would rate works by real artists higher in these scales that works by children or animals, and indeed their experiments validated their hypothesis.

We also hypothesized that human subjects would rate art by real artists higher on these scales than those generated by the proposed system. To our surprise the results showed that our hypothesis is not true! Human subjects rated the images generated by the proposed system higher than those created by real artists, whether in the Abstract Expressionism set or in the Art Basel set (see Table 3).

It might be debatable what a higher score in each of these scales actually means, and whether the differences are statistically significant. However, the fact that subjects found the images generated by the machine intentional, visually structure, communicative, and inspiring, with similar levels to actual human art, indicates that subjects see these images as art! Figure show several examples generated by CAN, ranked by the responses of human subjects to each question.

Discussion and Conclusion

We proposed a system for generating art with creative characteristics. We demonstrated a realization of this system based on a novel creative adversarial network. The system is trained using a large collection of art images from the 15th century to 21st century with their style labels. The system is able to generate art by optimizing a criterion that maximizes stylistic ambiguity while staying within the art distribution. The system was evaluated by human subject experiments which showed that human subjects regularly confused the generated art with the human art, and sometimes rated the generated art higher on various high-level scales.

What creative characteristics does the proposed system have? Colton 2008 suggested three criteria that a creative system should have: the ability to produce novel artifacts (imagination), the ability to generate quality artifacts (skill), and the ability to assess its own creation (Colton(2008)). Our proposed system possesses the ability to produce novel artifacts because the interaction between the two signals that derive the generation process is designed to force the system to explore creative space to find solutions that deviate from established styles but stay close enough to the boundary of art to be recognized as art. This interaction also provides a way for the system to self-assess its products. The quality of the artifacts is verified by the human subject experiments, which showed that subjects not only thought these artifacts were created by artists, but also rated them higher on some scales than human art.

One of the main characteristics of the proposed system is that it learns about the history of art in its process to create art. However it does not have any semantic understanding of art behind the concept of styles. It does not know anything about subject matter, or explicit models of elements or principle of art. The learning here is based only on exposure to art and concepts of styles. In that sense the system has the ability to continuously learn from new art and would then be able to adapt its generation based on what it learns.

We leave open how to interpret the human subjects' responses that ranked the CAN art better than the Art Basel samples in different aspects. Is it because the users have typical style-backward bias? Are the subjects biased by their aesthetic assessment? Would that mean that the results are

not that creative? More experiments are definitely needed to help answer these questions.

References

- Ellie Baker and Margo I Seltzer. Evolving line drawings. 1993.
- Daniel E Berlyne. Arousal and reinforcement. In *Nebraska symposium on motivation*. University of Nebraska Press, 1967.
- Daniel E Berlyne. *Aesthetics and psychobiology*, volume 336. JSTOR, 1971.
- Simon Colton. Creativity versus the perception of creativity in computational systems. In *AAAI spring symposium: creative intelligent systems*, volume 8, 2008.
- Simon Colton, Jakob Halskov, Dan Ventura, Ian Gouldstone, Michael Cook, and Blanca Perez-Ferrer. The painting fool sees! new projects with the automated painter. In *Proceedings of the 6th International Conference on Computational Creativity*, pages 189–196, 2015.
- Steve DiPaola and Liane Gabora. Incorporating characteristics of human creativity into an evolutionary art algorithm. *Genetic Programming and Evolvable Machines*, 10(2):97–110, 2009.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- Jeanine Graf and Wolfgang Banzhaf. Interactive evolution of images. In *Evolutionary Programming*, pages 53–65, 1995.
- Derrall Heath and Dan Ventura. Before a computer can draw, it must first learn to see. In *Proceedings of the 7th International Conference on Computational Creativity*, page to appear, 2016.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.
- Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.
- Penousal Machado, Juan Romero, and Bill Manaris. An iterative approach to stylistic change in evolutionary art.
- Colin Martindale. *The clockwork muse: The predictability of artistic change*. Basic Books, 1990.
- Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*. Retrieved June, 20:14, 2015.
- Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016.

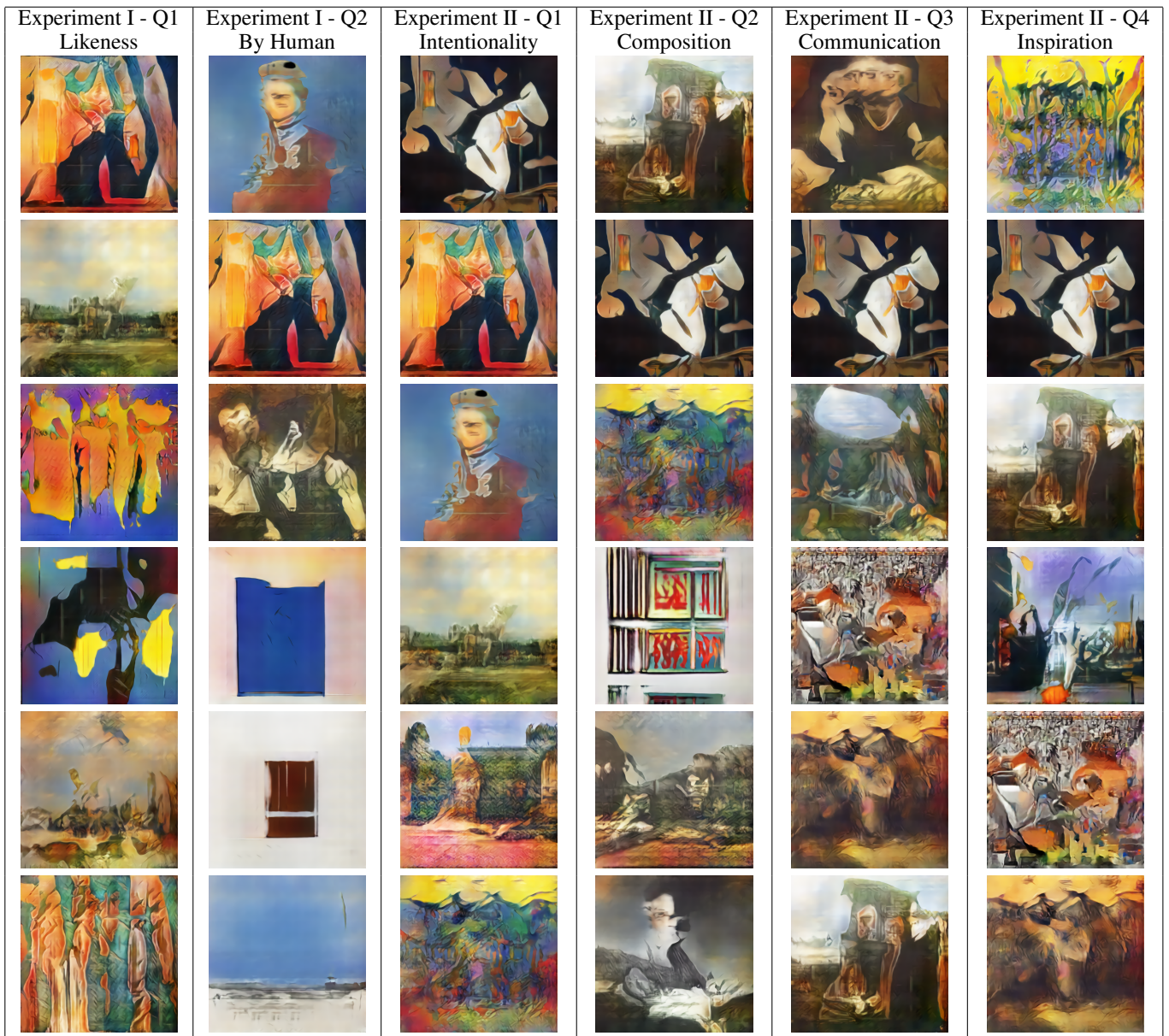


Figure 2: Top Ranked Images From CAN in Human Subject Experiment I and II

Scott Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

Theodore Christian Schneirla. An evolutionary and developmental theory of biphasic processes underlying approach and withdrawal. 1959.

Lior Shamir, Jenny Nissel, and Ellen Winner. Distinguishing between abstract art by artists vs. children and animals: Comparison between human and machine perception. *ACM Transactions on Applied Perception (TAP)*, 13(3):17, 2016.

Karl Sims. *Artificial evolution for computer graphics*, vol-

ume 25. ACM, 1991.

Leslie Snapper, Cansu Oranç, Angelina Hawley-Dolan, Jenny Nissel, and Ellen Winner. Your kid could not have done that: Even untutored observers can discern intentionality and structure in abstract expressionist art. *Cognition*, 137:154–165, 2015.

Wilhelm Max Wundt. *Grundzüge de physiologischen Psychologie*, volume 1. W. Engelman, 1874.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Human-Robot Co-Creativity: Task Transfer on a Spectrum of Similarity

Tesca Fitzgerald¹, Ashok Goel¹, Andrea Thomaz²

School of Interactive Computing, Georgia Institute of Technology¹

Department of Electrical and Computer Engineering, University of Texas at Austin²

{tesca.fitzgerald, goel}@cc.gatech.edu¹, athomaz@ece.utexas.edu²

Abstract

A *creative* robot autonomously produces a behavior that is novel and useful for the robot. In this paper, we examine creativity in the context of interactive robot learning from human demonstration. In the current state of interactive robot learning, while a robot may learn a task by observing a human teacher, it cannot later transfer the learned task to a new environment. When the source and target environments are sufficiently different, creativity is necessary for successful task transfer. In this paper we examine the goal of building creative robots from three perspectives. (1) Embodied Creativity: How may we ground current theories of computational creativity in perception and action? (2) Robot Creativity: How should a robot be creative within its task domain? (3) Human-Robot Co-Creativity: How might creativity emerge through human-robot collaboration?

Introduction

Robotics provides a challenging domain for computational creativity. This is in part because embodied creativity on a robotic platform introduces a dual-focus on agency and creativity. This is also partly because the robot's situatedness in perception and action in the physical world makes for high-dimensional input and output spaces. This results in several new constraints on theories of computational creativity: *autonomous* reasoning that responds to *high-dimensional*, real-world perceptual data to produce executable *actions* exhibiting a *creative behavior*. Additionally, it requires the robot to exhibit creativity in its *reasoning* as well as *physical* creativity due to its embodiment.

This distinction from other problems of computational creativity is especially evident in a robot that needs to transfer tasks learned in a familiar domain to novel domains. Each *task* consists of a series of *task steps* which are completed in sequence in order to produce the *task goal*. The goal of task transfer is to reuse the learned task steps in a manner that achieves the corresponding task goal in the new environment.

The topic of interactive robot task learning has been studied extensively (Argall et al. 2009; Chernova and Thomaz 2014). A common method for task learning involves the teacher providing the robot with a demonstration of the task, during which the teacher physically guides the robot's arm to

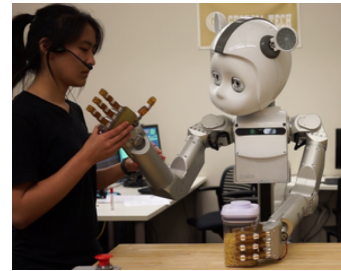


Figure 1: Interactive Task Demonstration

complete the task (as shown in Figure 1) (Argall et al. 2009; Akgun et al. 2012). The robot learns from this demonstration by recording the state of each degree-of-freedom in its arm at each time interval, recording the trajectory of its movement in order to train a model which can be used to repeat the task at a later time. Provided that a robot only learns of the task via a demonstration, its representation of that task is initially at the level of perception and action, and does not contain information about the high-level goals or outcomes of that task.

While a robot can learn to complete a task from demonstrations, it cannot immediately transfer the learned task model to perform the task in a new environment. For example, if objects in the new domain (referred to as the *target* domain) are configured differently than those in the original domain (the *source* domain), the robot may be able to apply the learned task model to the target domain *if* it has been parameterized according to the perceived locations of objects. However, if objects have been replaced in the target environment, the model is no longer parameterized based on the correct objects, and the robot cannot transfer the learned model. While a robot can be provided with additional demonstrations so that it generalizes over multiple instances of the task, this is a tedious and time-consuming task for the human teacher.

We address this problem of *task transfer*: transferring a task learned from one demonstration so that it can be reused in a variety of related target environments. As the previous example demonstrates, the difficulty of the task transfer problem increases as the source and target environments become more dissimilar. We propose the use of *human-robot co-creativity* to address difficult task transfer problems that

require the robot to perform a novel behavior. Just as creativity is evident in collaboration between humans (e.g. collaborating to assemble a structure out of blocks), human-robot co-creativity involves the coordination of novel, physical actions to achieve a shared goal. We present three perspectives on creative transfer: embodied creativity, robot creativity, and co-creativity. In doing so, we argue that:

- A robot exhibits creativity by (i) reasoning over past task knowledge, and (ii) producing a new sequence of actions that is different from the taught behaviors.
- For sufficiently difficult task transfer problems (in which the robot must produce an action that is different than that originally taught), creativity is necessary for the robot to perform task transfer successfully.
- Co-creativity occurs when the robot collaborates with the human teacher to perform task transfer, and is necessary in order to maintain autonomy while addressing a variety of transfer problems.

Related Work

Creativity in robotics is often discussed in the context of a robot performing behaviors that typically requires human creativity. Gemeinboeck & Saunders (2013) suggested that the embodiment of a robot lends it to be interpreted in the context of and in terms of human behaviors. The robot's enactment in human environments creates meaning to the observer. Gopinath & Weinberg (2016) explore the creative domain of musical robots and propose a generative model for a robot drummer to select natural and expressive drum strokes that are indistinguishable from a human drummer. Schubert & Mombaur (2013) model the motion dynamics that enables a robot to mimic creative paintings.

These are all examples of behaviors that appear novel to human observers and thus manifest social creativity. Bird & Stokes (2006) propose a different set of requirements of a creative robot: autonomy and *self-novelty*. The robot's solutions are novel to itself, regardless of their novelty to a human observer, thus manifesting personal creativity. Saunders, Chee, & Gemeinboeck (2013) address robot control in embodied creative tasks. In such domains, emphasis is placed on the result of the system, particularly how it enables co-creative expression when a human user interacts with it. Kantosalo & Toivonen (2016) propose a method for alternating co-creativity, in which the creative agent interacts with a teacher during a task, iteratively modifying the shared creative concept. Davis et al. (2015) describe *Drawing Apprenticeship*, which takes turns with a human artist to make drawings.

Colin et al. (2016) describe a creative process for reinforcement learning agents. Rather than focus on producing a creative output, they address the process of creativity by introducing a hierarchy of problem spaces, which roughly represent different abstractions of the original reinforcement learning problem. Vigorito & Barto (2008) also address creativity as a matter of creative process, rather than creative outcome. They address creative reasoning via a process that emphasizes (i) sufficient variation and (ii) sufficient selection of candidate policies. In addressing the first, they propose variation by representing the problem at multiple levels

of abstraction. They propose that new behaviors can only be discovered by representing the learning problem (and thus the search space) at a sufficient abstraction such that steps through the space explore a range of variations. By stepping through the search space at one of many levels of abstraction, solutions can be explored which would not be accessible by searching through the space at a lower level of abstraction.

We build off this distinction between creative robots which (i) produce novel output, and/or (ii) reason creatively. Particularly, we argue that a robot which suitably addresses the problem of creative transfer must exhibit creativity in both regards, while also meeting a third criteria of *autonomy*: performing task transfer with as little input from the human teacher as necessary.

Case-based reasoning provides one conceptual framework for exploring task transfer in interactive robotics (Kolodner 1993; Goel and Díaz-Agudo 2017). Analogical reasoning provides another, more general framework (Gentner and Markman 1997; Falkenhainer, Forbus, and Gentner 1989; Gick and Holyoak 1983; Thagard et al. 1990). In analogical reasoning, the difference between source and target problems may lie on a spectrum of similarity (Goel 1997). At one end of this spectrum, the target problem may be identical to the source problem so that memory of the source problem directly supplies the answer to the target. At the other extreme of the similarity spectrum, the target problem is so different from the source problem that transfer between the two is not feasible. In between the two extremes, transfer entails problem abstraction where the level of abstraction may depend on the degree of similarity between the source and target problems (Goel and Bhatta 2004). Oltețeanu & Falomir (2016) describe a method for object replacement, enabling creative improvisation when the original object for a task is unavailable. Fauconnier & Turner (2008) introduced *conceptual blending*: a tool for addressing analogical reasoning and creativity problems, obtaining a creative result by merging two or more concepts to produce a new solution to a problem. Abstraction is enabled by mapping the merged concepts to a *generic space*, which is then grounded in the *blend space* by selecting aspects of either input solution to address each part of the problem. Applied to a robotic agent which uses this creative process to approach a new transfer problem, the robot may combine aspects of several learned tasks to produce a new behavior.

Transfer as a Creativity Problem

In Related Works, we have identified two criteria commonly applied to creative robots: (i) autonomy, and (ii) production of novel output, and/or utilization of a creative reasoning process.

Autonomy Rather than rely on receiving a new demonstration of the entire task, an autonomously creative robot must reason about the task using the representation it has previously learned, while also minimizing its reliance on the human teacher. We claim that this criteria does not preclude

the robot from deriving new information from human interaction, provided that (i) the robot does not require a full re-demonstration of the task, and (ii) the robot reasons over what information is needed from the teacher and how to request that information. We refer to a robot that meets these two criteria while collaborating with a human teacher as exhibiting *partial-autonomy*.

Novel output The robot learns to complete a task with respect to the locations of relevant objects (e.g. pouring is an action which is completed with respect to the location of a bowl and a scoop). By parameterizing the skill models (learned from the demonstration) based on object locations, simple adjustments can be made to objects' locations without altering the skill model itself. However, once a transfer problem requires significant changes to the skill model (either in constraints of the model, or a replacement of the model entirely), it no longer produces the same action. The revised model is reflective of a behavior that is both novel to the human teacher (since it is different than what was taught), and novel to the robot (since it is distinct from the output of other skill models the robot may have recorded).

Creative reasoning A robot may need to derive additional information about the task in the target environment. By interacting with a human teacher to request additional task information, the robot would leverage *co-creativity* in which the robot and human teacher collaborate to produce a novel result. As an alternate approach, a robot can address a target environment by combining aspects of its previous experiences. For example, a robot may know how to pour a mug, and separately, how to pick up a bowl. Knowledge of these two tasks may be combined in order to address a new problem, such as the robot needing to pour a bowl. By performing *conceptual blending* in this way, the robot would leverage a creative reasoning process.

Perspectives on Creative Transfer

We now introduce three perspectives on the problem of creative transfer: embodied creativity, robot creativity, and co-creativity. Each of these perspectives highlights a different challenge of the creative transfer problem.

Embodied Creativity

Systems of *embodied creativity*, such as the creative robot we have discussed, introduce challenges as a result of their embodiment. Specifically, the input that is available to the embodied agent and the output that must be produced are at a level of detail that reflects how the agent can perceive or act in the physical world.

Input and Output Requirements An example of this type of input is an agent's perception of its environment using a 3D RGBD camera. This provides the agent with a point-cloud representation of its environment, and can be segmented to identify features of each object (e.g. dimensions, location, color histogram) using methods such as described in (Trevor et al. 2013). Figure 3 depicts an overhead view of a robot's table-top environment, and the corresponding object segments observed by the robot.

In an robot which learns from task demonstrations, the human teacher manually guides the robot's hand (end-effector) to complete a task. During this demonstration, it may record both (i) the position of each joint in its arm, and (ii) the 6D cartesian pose (x, y, z, roll, pitch, yaw) of its end-effector. These recordings are measured at each time interval, resulting in a trajectory of the robot's arm or end-effector positions over time.

A *skill model* can then be trained on this trajectory, such that a similar motion can be repeated at a later time. Many skill models have been proposed which encode the task demonstration and are used to plan a motion trajectory reproducing the task at a later time (Chernova and Thomaz 2014; Argall et al. 2009; Akgun et al. 2012; Pastor et al. 2009; Niekum et al. 2012; Bruno, Calinon, and Caldwell 2014). Should the robot receive multiple demonstrations of the task, the skill model provides a generalization over the full set of demonstrations. Object locations are used to parameterize the skill model, so that differences in object locations in the target environment can be accounted for by using segmented object features as parameters.

The agent's embodiment also enforces a specific output type: a motion trajectory which reproduces the task in the target environment. This trajectory must indicate the position of each joint at each time interval, over the entire course of the task.

Role of Embodiment in Creativity We propose that the role of embodiment in creativity can be expressed on a spectrum. At one end of the spectrum, embodiment plays no role in the creative process until the creative result is to be executed on the robot. Systems which perform in a creative domain (e.g. Schubert and Mombaur 2013) typically operate at this level, where the emphasis is on engaging in creative domains that exist in the physical world (and thus must be executed by an embodied agent). At the other end of the spectrum, the embodiment is an integral element of the creative model. Creative reasoning is performed with respect to the constraints of embodiment. Intermediate methods have been proposed, where the embodiment is modeled alongside, but separately from, the creative task (e.g. Gopinath and Weinberg 2016).

In previous work (Fitzgerald, Goel, and Thomaz 2015), we have defined the Tiered Task Abstraction (TTA) representation for tasks learned from demonstrations. This representation is intended to perform creative transfer by integrating the agent's embodiment into the task representation itself. The TTA representation contains the following elements:

- **Skill Models:** The task demonstration is segmented into *task steps*, each of which is represented by a separate *skill model*. These models are parameterized in terms of a start and end location, while maintaining the trajectory "shape" of the demonstrated action.
- **Parameterization Functions:** These reflect constraints which guide the start and end position of each task step as an offset from an object location. For example, scooping ends with the robot's end-effector 5 cm above the pasta

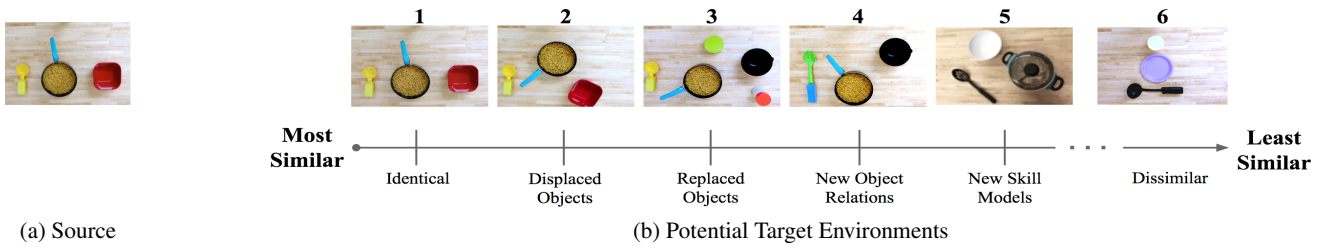


Figure 2: Spectrum of Similarity Between Source and Target Environments

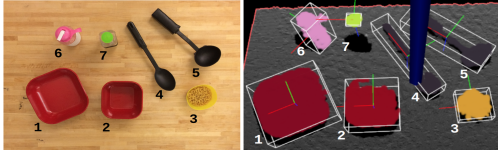


Figure 3: An overhead view of a table-top environment (left) and the segmented point cloud representation (right)

bowl, before continuing with the next task step. The corresponding parameterization function is: $\langle o_x, o_y, o_z + 5 \rangle$, where o is a reference to the relevant object (in this case, the location of the pasta bowl).

- **Object Labels:** These are the labels which are uniquely associated with each object instance identified in the environment. Each labeled object represents a single object which is consistent over a range of feature values.
- **Object Features:** These are the feature values associated with each object label. While the label represents a static object, the specific feature values may differ depending on the environment, e.g. object locations, color (based on lighting conditions), spatial configurations, and properties.

Note that each element is parameterized by the next; by omitting one or more elements from the task representation, the resulting representation is one that is *abstracted*. In doing so, a task can be represented at a level of abstraction which is common to both the source and target environments. However, once a representation is abstracted, it must be *grounded* in the target environment in order to produce an output which is executable by the robot. In an embodied system, grounding refers to parameterizing a representation based on perception in the physical world. A representation is *grounded* in a target environment when each of its elements (skill models, parameterization functions, object labels, and object features) are present and defined based on information derived in the target environment (either by perception or interaction in the target environment). This challenge of abstraction and grounding is at the core of embodied creativity.

Robot Creativity

Related to an embodied, creative agent, a *creative robot* must also account for issues of embodiment (e.g. input from real-world perception and output as an executable trajectory). We now address additional challenges which result

from robot domains, particularly the types of *tasks* which a robot may be expected to perform. Given enough demonstrations of a task, a robot can learn a model which generalizes across them, enabling it to address target environments which are similar to the source environments it has observed. However, this introduces several constraints:

1. The human teacher must be able to provide several demonstrations of the task, which can be time-consuming and tedious.
2. The teacher must know what target environments the robot is likely to address, so that similar source environments can be selected for demonstrations.
3. The robot is still limited to addressing target environments which are closely similar to the observed source environments.

While providing more demonstrations does increase the model’s generalizability, these constraints still apply. This precludes many opportunities for addressing realistic transfer problems, in which the robot needs to make broader generalizations. Examples of such tasks include stacking plates after learning to stack wood blocks, or pouring a coffee pot after learning to pour a cup. Without a representation of the relation between objects in the source and target environments, the robot is unable to parameterize its task model based on the correct objects in the target environment. Furthermore, more difficult transfer problems are also plausible, such as tasks in which new constraints are added in the target environment which could not be learned in the source environment.

Task Similarity Spectrum In previous work (Fitzgerald, Goel, and Thomaz 2015), we have discussed task transfer as a problem which ranges in the similarity between the source and target environments. The outcome of this is that task transfer problems may vary in difficulty. While we will argue that some categories of task transfer do require a co-creative approach, task transfer does not inherently necessitate creativity. For example, a task demonstrated in a source environment (e.g. Fig. 2a) can be directly reused in a target environment which either (i) does not require modification of the learned task (image 1 in Fig. 2b), or (ii) requires parameterization based on object location (image 2 in Fig. 2b), provided that it has been parameterized according to the locations of objects. Since the learned skill models are reused to address these transfer problems (albeit, modified to account for new object locations), the outcome is

	Identical Problem	Displaced Objects	Replaced Objects	New Object Relations	New Skill Models
Retained Knowledge	Goal Skill models Param functions Object labels Object features	Goal Skill models Param functions Object labels	Goal Skill models Param functions	Goal Skill models	Goal
Grounded Knowledge	None	Object features	Object labels Object features	Param functions Object labels Object features	Skill models Param functions Object labels Object features

Figure 4: Summary of retained and grounded elements at each level of abstraction

novel to neither the robot nor the human teacher, and thus is not an example of creativity. Similarly, in transferring a task to a target environment which requires an object mapping (image 3 in Fig. 2b), the original skill model can still be reused; prior to parameterizing it according to object locations, the robot must first obtain a *mapping* between objects in the source and target environments. With this mapping, the skill model can be re-parameterized according to the correct objects. Again, the learned skill models are reused (this time after applying an object mapping and re-parameterizing the skill models), and so the resulting action is not novel to the robot or human teacher.

In contrast to these three examples, consider target environments 4 and 5 in Figure 2b. Figure 4 differs from the source in Figure 2a in that objects are: (i) displaced, (ii) replaced, and now (iii) *constrained* because of the new scoop size. The robot’s actions must now be constrained such that its end-effector remains higher above the table in order to complete the task successfully. The skill model parameters, which reflect constraints of the task by indicating the relation between the robot’s end-effector and object locations, cannot be directly reused in this target environment. In order to address this problem, new *parameterization functions* must be identified in the target environment, applying constraints to the learned skill models that are distinct from those of the original demonstration. Provided that a robot can identify the new parameterization functions with some degree of autonomy (e.g. does not simply receive a new demonstration of the task in the target environment), this category of transfer problems meets the criteria for creative transfer: partial-autonomy and novel output.

Target 5 in Figure 2b differs from the source in similar respects, with one additional difference: an extra step is needed in order to lift the lid off the pasta pot prior to scooping the pasta. As a result, the original skill models learned in the source cannot be directly transferred. In addition to deriving new parameterization functions in the target environment, this problem also requires that the robot derive or learn a new skill model to account for the missing step. In a later section, we discuss potential methods for deriving this information via further interaction with the human teacher; however, regardless of what method is used, the robot (i) autonomously transfers the task representation (since it does not rely on receiving a full re-demonstration of the task), (ii) produces action that is novel to both the robot and the hu-

man teacher, and (iii) utilizes a creative reasoning method (by blending previously and newly learned skill models). Therefore, a robot that successfully completes transfer problems of this kind meets the criteria for creativity.

These task differences illustrate a *spectrum* of similarity between the source and target; at one end of the spectrum, the source and target differ in small aspects such as object configurations. At the other end of the spectrum, they contain more differences, until finally (as in target 6), the target environment cannot be addressed via transfer. While we have highlighted discrete levels of similarity in this spectrum, we do not claim this to be an exhaustive categorization of transfer problems. Figure 2 illustrates that without addressing problems of creative transfer, task transfer methods are limited to addressing a narrower set of transfer problems: those in which the target environment does not require novel behavior or reasoning to address. By examining problems of creative transfer, we broaden the range of problems that a robot can address from transferring a single task demonstration.

Transfer Via Task Abstraction In previous work, we have found that as the source and target environments become more dissimilar (according to the similarity spectrum in Fig. 2), the task must be represented at increasing levels of abstraction for transfer to be successful (Fitzgerald, Goel, and Thomaz 2015). We have summarized these task differences in Figure 4. For problems of non-creative transfer, we have also demonstrated that the abstracted representation can be grounded through perception (e.g. by completing the *object features* element based on perception of the target environment) and/or interaction with the human teacher (e.g. by using interaction with the teacher in the target environment to infer the *object labels* element).

To address problems in which objects are displaced in the target environment, the *object features* element must be grounded in the target environment, while other elements of the original representation can be retained. This grounding occurs by observing the new object locations in the target (Pastor et al. 2009; Fitzgerald, Goel, and Thomaz 2015).

To address problems in which objects are replaced in the target environment, both the object features and *object labels* must be grounded in the target environment. We have demonstrated a method for grounding this information by inferring an *object mapping* from guided interaction with the human teacher (Fitzgerald et al. 2016). An object map-

ping indicates which objects in the source environment correspond to each object in the target environment, and is used to ground object labels in the target environment. By asking the teacher to assist in the object mapping by indicating the first object the robot should use in the target environment, the robot can attempt to infer the remainder of the object mapping.

To similarly abstract and ground the task representation in order to address problems of creative transfer (including problems in the New Object Relations and New Skill Models categories), two elements of the TTA representation must be grounded in the target environment: the parameterization functions (for both categories of creative transfer problems) and skill models (for creative transfer problems involving new skill models). This is a challenge because these two elements cannot be directly observed via perception (as was possible when grounding object features) and cannot be inferred (as was possible when inferring an object mapping). Rather, they are dependent on knowledge of the goal of the task, which the robot does not have. We next discuss interactive solutions to challenge by taking a co-creative perspective on creative transfer.

Co-Creativity

In the context of an embodied robot which is situated in a task domain, a robot may continue to interact with a human teacher during task transfer. Thus, the robot may leverage the human teacher’s knowledge of the task domain in order to engage in a co-creative transfer process.

As discussed in the previous section, the robot required little assistance in order to address problems of non-creative transfer. The first two categories of transfer problems (e.g. identical and displaced-objects environments) could be addressed by the robot with full autonomy. The third category of transfer problems (e.g. replaced-objects environments) required some assistance from the human teacher in order to indicate which objects the robot should use in the first few steps of the task.

In order to address problems of creative transfer, the robot must ground the (i) parameterization functions and (ii) skill models in the target environment. These are the two elements of the TTA representation which contain the most high-level information about the task: the constraints between the robot’s hand and objects in the environment, and the skill model which preserves the trajectory shape of the demonstrated action, respectively. Because these represent high-level information and are informed by the goal of the task, they cannot be grounded by the robot with complete autonomy. Presuming that the human teacher is aware of the goal of the task, and how that goal should be met in the target environment, we posit that the teacher is available to assist the robot in reaching that goal. It is advantageous for the robot to continue to interact with the human teacher in order to ground these representation elements, since the teacher does know how the task should be performed to achieve the task goal. The aim of this *co-creative* approach is to produce a solution that (i) is partially autonomous (the robot interacts with a human teacher and may receive additional instruction, but does not require a full re-demonstration of the task),

(ii) enables collaboration with the human teacher so that the robot may infer information about the task in the target environment, (iii) results in parameterization functions and/or skill models that can ground an abstracted task representation, and (iv) grounds the TTA representation such that a trajectory can be executed in the target environment.

Figure 4 summarizes the representation elements which must be retained or grounded for each category of transfer problems. This relation between (i) task similarity and (ii) assistance from the human teacher introduces a second dimension to the aforementioned similarity spectrum; as the source and target environments become more dissimilar, the robot’s level of transfer autonomy decreases and its dependence on interaction with the human teacher increases. We now discuss two forms of interaction for human-robot co-creativity.

Grounding Parameterization Functions In order to address problems in the New Object Relations category, three representation elements must be grounded: object features, object labels, and parameterization functions. In previous work (Fitzgerald et al. 2016), we demonstrated a simulated robot asking for assistance to identify the object mapping between objects in the source and target environments. In implementing this system on a physical robot, a robot could request assistance after each step of the task by asking “What do I use next?”, to which the teacher would respond by handing the robot the next object involved in the task. Each assistance would provide a single correspondence (e.g. the red bowl is mapped to the blue bowl). Additional assistance would be derived by asking the teacher where to place the object, to which the teacher would respond by pointing at the next goal location. After each hint, the remainder of the object mapping (e.g. the mapping of objects for which the robot has not yet received assistance) would be predicted by calculating mapping confidence after each assistance.

Similarly, when grounding parameterization functions, the robot should interact with the teacher so that it infers the necessary information to ground missing elements of the task representation, without requiring too much information and time from the human teacher (so as to maximize the robot’s autonomy). We propose a method for grounding parameterization functions in a manner similar to object mapping. Rather than evaluate only the object mapping confidence at each step of the task, the robot should also verify its confidence in using the next step’s parameterization function. One method of measuring confidence may be to compare the objects used in the next step to those which the robot would have used in the source environment. Assuming that similarly-shaped objects can be manipulated in similar ways, dissimilar objects may need to be manipulated differently despite serving the same purpose. Oltețeanu & Falomir (2016) proposed a method for identifying the suitability of object replacements in simulation, based on features such as shape and affordances. We expect that similar features will play a role in evaluating the robot’s confidence in using a novel object, and must be extracted from a physical robot’s perception (similar to how object features were obtained in Fitzgerald et al. 2016). If the robot is not confi-

Algorithm 1 Grounding Parameterization Functions

```
1: function GROUNDPARAMFUNCTIONS(S)
2:    $map \leftarrow$  empty mapping
3:   while target task is incomplete do
4:     if  $map$  is incomplete then
5:        $h \leftarrow$  next mapping hint from teacher
6:        $map \leftarrow map + \text{PredictMapping}(h)$ 
7:     end if
8:      $s \leftarrow \text{GetNextStep}(\text{source demo } C_s)$ 
9:      $o_n \leftarrow \text{GetNextObject}(s, map, \text{target objects } O_t)$ 
10:    if  $\text{ObjectSim}(o_n, \text{source objects } O_s) < \beta$  then
11:      ask teacher to reposition end-effector
12:       $r \leftarrow$  record end-effector displacement from
        nearest object
13:       $\text{SetParamFunction}(s, r)$ 
14:    end if
15:     $\text{ExecuteNextStep}(s)$ 
16:  end while
17: end function
```

dent in this similarity (meaning its confidence value is below some threshold β), it can request the human teacher to align its end-effector in preparation to complete the next step of the task. The robot would then record the parameterization function as an offset from the closest object. Algorithm 1 outlines this process.

Grounding Skill Models To address tasks requiring new skill models (such as the final target environment image in Figure 4), the robot will need to ground the same elements as before (object features, object labels, and parameterization functions) in addition to the new skill models. To do this, we hypothesize that the robot can again evaluate its confidence for completing each step of the task. We introduce an additional threshold to this evaluation process: if object similarity is below a second threshold α (such that $\alpha < \beta$), then the robot searches for other previously-learned task demonstrations which contain the unfamiliar object. If there exists another demonstration using the same object, the robot should then evaluate the similarity between (i) the task step involving the object in the original source environment and (ii) the task step in the newly-retrieved demonstration that involves the new object. If the two task steps appear similar, then the newly-retrieved task step may be an alternate version of the step adapted for that object, and can be applied toward reproducing the task in the target environment. If they are not similar, then the robot may ask the teacher to re-demonstrate that particular step of the task. Algorithm 2 outlines this process.

Directions for Continued Work

We have introduced three perspectives on the problem of creative transfer. *Embodiment* introduces challenges of perception and action which must be integrated into the creative process. The domains that a *creative robot* encounters adds additional constraints; we have argued that for some categories of task transfer problems, creativity is necessary for the robot to transfer past task knowledge and produce a new

Algorithm 2 Grounding Skill Models

```
1: function GROUNDSKILLMODELS(S)
2:    $map \leftarrow$  empty mapping
3:   while target task is incomplete do
4:     if  $map$  is incomplete then
5:        $h \leftarrow$  next mapping hint from teacher
6:        $map \leftarrow map + \text{PredictMapping}(h)$ 
7:     end if
8:      $s \leftarrow \text{GetNextStep}(\text{source demo } C_s)$ 
9:      $o_n \leftarrow \text{GetNextObject}(s, map, \text{target objects } O_t)$ 
10:    if  $\text{ObjectSim}(o_n, \text{source objects } O_s) < \beta$  then
11:      find a demo with step  $s_{new}$  containing  $o_n$ 
12:      if  $\text{ActionSimilarity}(s_{new}, s) < \alpha$  then
13:        ask teacher to demonstrate next step
14:         $a \leftarrow$  record demonstrated task step
15:         $r \leftarrow$  record end-effector displacement
        from nearest object
16:         $\text{SetSkillModel}(s, \text{TrainSkillModel}(a))$ 
17:         $\text{SetParamFunction}(s, r)$ 
18:      else
19:         $s \leftarrow s_{new}$ 
20:      end if
21:    end if
22:     $\text{ExecuteNextStep}(s)$ 
23:  end while
24: end function
```

action which is different from the originally taught behaviors. By interacting with the human teacher to produce a result which is both (i) distinct from that of the original task demonstration and (ii) achieved through a combination of the robot’s reasoning and the teacher’s assistance, the robot and human teacher use a *co-creative* process to address the task transfer problem. This enables the robot to leverage the teacher’s knowledge of the task goals and how they are achieved in the target environment, while also minimizing the time required of the human teacher to provide assistance.

We propose several directions for continued work on co-creative transfer. First, we hypothesize that there are several alternative approaches to interactive task grounding. For example, the robot may use speech as the assistance modality by asking about objects prior to attempting to perform the task. Alternatively, the robot could instead rely on the teacher to correct its actions (rather than proactively ask for assistance) after each task step. Transfer problems of increased difficulty may be also addressed via exploration, in which the robot collaborates with the human teacher to creatively explore new actions, to which the human teacher can respond by guiding the robot’s exploration. Second, we have proposed two algorithms for co-creative transfer, and suggest that future work should implement these on a physical robot. This will also engender questions of interaction; how should the robot request specific types of assistance from the teacher? We expect that the implementation of this will result in additional questions of how the robot should behave in order to best leverage the teacher’s knowledge. Finally, we have identified two categories of creative transfer prob-

lems, and associated each with a task abstraction which can be used to address problems in these categories. However, we do not claim this to be an exhaustive list of creative transfer problem categories. We propose an area of continued work to identify other applications of creative task transfer, which may occur in problems which require more creativity to address. We suggest that further work on creative transfer explore the dimensions along which a creative transfer problem becomes more (or less) difficult.

Acknowledgments

This material is based on work supported by the NSF Graduate Research Fellowship under Grant No. DGE-1148903.

References

- Akgun, B.; Cakmak, M.; Jiang, K.; and Thomaz, A. L. 2012. Keyframe-based learning from demonstration. *International Journal of Social Robotics* 4(4):343–355.
- Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.
- Bird, J., and Stokes, D. 2006. Evolving minimally creative robots. In *Proceedings of the Third Joint Workshop on Computational Creativity*, 1–5. IOS Press, Amsterdam.
- Bruno, D.; Calinon, S.; and Caldwell, D. G. 2014. Learning adaptive movements from demonstration and self-guided exploration. In *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*, 101–106. IEEE.
- Chernova, S., and Thomaz, A. L. 2014. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(3):1–121.
- Colin, T. R.; Belpaeme, T.; Cangelosi, A.; and Hemion, N. 2016. Hierarchical reinforcement learning as creative problem solving. *Robotics and Autonomous Systems* 86:196–206.
- Davis, N.; Hsiao, C.-P.; Singh, K. Y.; Li, L.; Moningi, S.; and Magerko, B. 2015. Drawing apprentice: An enactive co-creative agent for artistic collaboration. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 185–186. ACM.
- Falkenhainer, B.; Forbus, K. D.; and Gentner, D. 1989. The structure-mapping engine: Algorithm and examples. *Artificial intelligence* 41(1):1–63.
- Fauconnier, G., and Turner, M. 2008. *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books.
- Fitzgerald, T.; Bullard, K.; Thomaz, A.; and Goel, A. 2016. Situated mapping for transfer learning. In *Fourth Annual Conference on Advances in Cognitive Systems*.
- Fitzgerald, T.; Goel, A.; and Thomaz, A. 2015. A similarity-based approach to skill transfer. In *Workshop on Women in Robotics at Robotics: Science and Systems*.
- Gemeinboeck, P., and Saunders, R. 2013. Creative machine performance: Computational creativity and robotic art. In *Proceedings of the 4th International Conference on Computational Creativity*, 215–219.
- Gentner, D., and Markman, A. B. 1997. Structure mapping in analogy and similarity. *American psychologist* 52(1):45.
- Gick, M. L., and Holyoak, K. J. 1983. Schema induction and analogical transfer. *Cognitive psychology* 15(1):1–38.
- Goel, A. K., and Bhatta, S. R. 2004. Use of design patterns in analogy-based design. *Advanced Engineering Informatics* 18(2):85–94.
- Goel, A., and Díaz-Agudo, B. 2017. What's hot in case-based reasoning? In *Proceedings of the Thirty-First AAAI Conference*.
- Goel, A. K. 1997. Design, analogy, and creativity. *IEEE expert* 12(3):62–70.
- Gopinath, D., and Weinberg, G. 2016. A generative physical model approach for enhancing the stroke palette for robotic drummers. *Robotics and Autonomous Systems* 86:207–215.
- Kantosalo, A., and Toivonen, H. 2016. Modes for creative human-computer collaboration: Alternating and task-divided co-creativity. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Kolodner, J. 1993. *Case-based reasoning*. Morgan Kaufmann.
- Niekum, S.; Osentoski, S.; Konidaris, G.; and Barto, A. G. 2012. Learning and generalization of complex tasks from unstructured demonstrations. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 5239–5246. IEEE.
- Oltețeanu, A.-M., and Falomir, Z. 2016. Object replacement and object composition in a creative cognitive system. towards a computational solver of the alternative uses test. *Cognitive Systems Research* 39:15–32.
- Pastor, P.; Hoffmann, H.; Asfour, T.; and Schaal, S. 2009. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 763–768. IEEE.
- Saunders, R.; Chee, E.; and Gemeinboeck, P. 2013. Evaluating human-robot interaction with embodied creative systems. In *Proceedings of the fourth international conference on computational creativity*, 205–209.
- Schubert, A., and Mombaur, K. 2013. The role of motion dynamics in abstract painting. In *Proceedings of the Fourth International Conference on Computational Creativity*, volume 2013. Citeseer.
- Thagard, P.; Holyoak, K. J.; Nelson, G.; and Gochfeld, D. 1990. Analog retrieval by constraint satisfaction. *Artificial Intelligence* 46(3):259–310.
- Trevor, A. J.; Gedikli, S.; Rusu, R. B.; and Christensen, H. I. 2013. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*.
- Vigorito, C. M., and Barto, A. G. 2008. Hierarchical representations of behavior for efficient creative search. In *AAAI Spring Symposium: Creative Intelligent Systems*, 135–141.

Blend City, BlendVille

João Gonçalves, Pedro Martins, Amílcar Cardoso

CISUC, Department of Informatics Engineering

University of Coimbra, Portugal

{jgconc,pjmm,amilcar}@dei.uc.pt

Abstract

This paper presents BlendVille, a computational system based on the Conceptual Blending framework, where the search for the best blend is handled as an optimisation task by a Multi-Threaded (MT) Genetic Algorithm (GA). The new system departs from ideas explored in a previous framework, Divago. One of the most substantial differences from the latter lies in the usage of entropy to create new concepts with varying levels of information complexity. Additionally, Blendville explores the use of multiple analogies when projecting concepts into the blend space and in the evolutionary process. We investigate the behaviour of the new system, compare its output with its predecessor's and report on our findings.

Introduction

The Conceptual Blending (CB) theory (Fauconnier and Turner 2002) was proposed to explain mechanisms involved in the creation of meaning and insight in the every day mind. In the last years, one can witness an emergence of various computational systems based on CB with diverse origins, including international projects such as CoInvent (Schorlemmer et al. 2014) and ConCreTe (Žnidaršič et al. 2016).

This paper describes recent efforts, initiated within the latter project, towards the proposal of a new, written from scratch, computational approach to CB, which we named *BlendVille*. We build on the legacy of *Divago*, one of the first and most comprehensive implementations of CB, developed by Pereira (2005).

We start this paper with short overviews of both the Conceptual Blending theory and the Divago computational framework. Then, we expose our proposed blender, its inner workings and the optimality measures the system uses, after which we evaluate the impact of those measures on the system's output. Finally, we outline further work to improve our blender and conclude on our findings.

Background

Conceptual Blending (CB) was suggested as cognitive theory by Fauconnier and Turner (2002) to explain

processes of conceptual integration occurring in human thought. Its potential to model mechanisms of concept invention has increasingly inspired research in Computational Creativity in recent years (e.g., the recent works by (Žnidaršič et al. 2016) in the ConCreTe project and (Schorlemmer et al. 2014) in the CoInvent project). A key element in the theory is the *mental space*, a partial and temporary structure of knowledge assembled for purposes of thought and action (Fauconnier and Turner 2002). The CB process takes two *input spaces* and looks for a partial *mapping* between elements of both spaces that may be perceived as similar or analogous in some respect. A third mental space, called *generic*, encapsulates the conceptual structure shared by the input spaces, providing guidance to the next step of the process, where elements from each of the input spaces are *selectively projected* into a new mental space, called the *Blend Space*. Further stages of the process elaborate and complete the blend.

As the input spaces can be blended in many forms, CB proposes a set of optimality principles to characterise good blends. These principles have a key role in the process and help to ensure that the resulting blend represents a coherent and integrated structure.

Another important notion is that of *frames*, which in the theory play a role in the structuring of mental spaces (Fauconnier and Turner 2002). Frames are mental structures that provide a kind of abstract prototyping of entities, actions or reasonings and may guide the process of blend construction to recognizable wholes (Pereira 2005). For instance, the mental space *Disney's Dumbo* encloses the idea of an elephant capable of flying, in which two invoked frames of thought are the frames *elephant* and *flight*.

Divago

Divago (Pereira 2005) is one of the first developed computational architectures based on the CB framework. It is composed of three major working modules (Fig. 1): the *Mapper*, the *Blender* and the *Factory*.

In Divago, input spaces are represented as computational versions of Conceptual Maps, i.e. graphs where nodes are concepts and arcs are relations between a source and a target concepts. The input spaces are

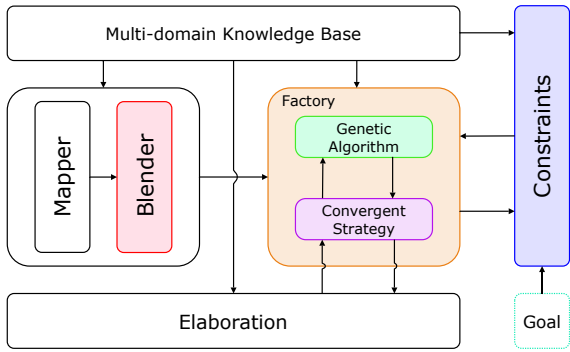


Figure 1: The architecture of Divago. The relevant modules to this paper are shown in colour. Best viewed in colour.

stored as semantic networks in the form of text triples.

The *Mapper* computes analogy mappings between concepts from two input spaces, using a structural alignment algorithm based on Sapper (Veale and Keane 1997). The input spaces are in the form of semantic networks, same structure used by our blender. The mappings calculated are grouped in a single set and represent the analogy of concepts to be used by the Blender module. The analogy is formed by finding the largest (in cardinality) isomorphic sub-graphs of both input spaces. Once the isomorphism is found, aligned pairs of nodes from both subspaces are mapped together to produce an analogy mapping.

The *Blender* takes one analogy and performs a selective projection into the blend space, which leads to the construction of a *blendoid*, an intermediate graph that subsumes the set of all possible blends. This blendoid feeds a GA in the *Factory* module, which explores the space of all possible combinations of projections of the input spaces taking into account the generic space.

The *Factory* uses an implementation of CB optimality principles, intended to ensure a coherent and integrated blend, as fitness function of the evolutionary process (the *Constraints* module). When an adequate solution is found, the *Factory* stops the execution and returns the best blend.

A city of blends

The architecture of BlendVille is shown in Fig. 2, where the search for the best blend is handled as an optimisation task by a GA that evolves a population of competing blends. By comparing with Fig. 1, we can see that BlendVille roughly plays the role of the modules *Factory* and *Constraints* in Divago. Its inputs are (i) a list of input spaces in the same format as Divago, (ii) a list of analogies and (iii) a list of frames. The fitness of each blend is computed through a weighted sum of a number of measures, discussed later. The reason for using a GA is that the space to search for an optimal blend is highly complex. This is mainly due to the high variability in the semantic structure of a blend

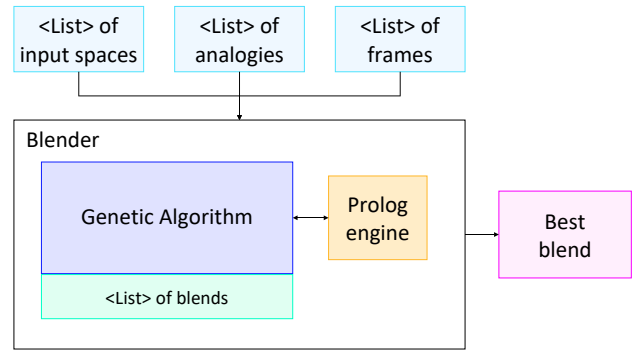


Figure 2: BlendVille's architecture and input knowledge. Best viewed in colour.

(relations, edge directions and concepts).

Three major differences between BlendVille and the Divago's *Factory* are: the usage of multiple analogies (sets of mappings) in the projection of concepts from the input spaces to the blend space; the investigation of different metrics/constraints for guiding the blending process; and the performance improvement of the blending process itself. The first two are discussed on a section of their own, further below.

To ensure a better performance, BlendVille uses a Multi-Threaded (MT) GA, allowing the evolution of a greater number of blends, the handling of complex semantic structures (input spaces, analogies, frames) and the parallel execution of multiple Prolog calls, used to check frame matching.

Input Spaces

The input spaces are represented as semantic graphs of concepts and relations (concept maps). Each input space graph represents a directed pseudo-graph, described as a set of triples $relation(concept_{source}, concept_{target})$, where both relations and concepts are text strings (arrays of characters). Our blender supports multiple input spaces, identified by their title (*horse*, *bird*, *boat*, etc.) which we term *domain* or *namespace*. In a declarative language (Prolog), the input space corresponds to a knowledge base of facts, where each fact is an edge of the graph, the predicate is the label of this edge (relation) and both concepts are the atoms of the fact. As an example, the sentence "Socrates is a man" corresponds to the fact $isa('Socrates', 'man')$ and in the semantic graph is represented as an edge labelled *isa* going from the vertex *Socrates* to the vertex *man*.

Analogies

Also required is a set of analogies. Each analogy relating two input spaces is defined as a set of mappings $m_i = \{c_i, c_j\}$, each of which associating two concepts c_i and c_j , **one from each space**. In each analogy, a concept can only be present in one mapping. However, a concept can be mapped to different concepts in different analogies. We expect that this rationale will favour a

higher blend diversity, when compared to a blender system which uses a single set of mappings (one analogy) as in Divago. Currently, we use an additional tool with an algorithm similar to Divago’s Mapper to generate analogies from the input spaces.

Frames

Frames are handled in the form of either semantic graphs or logical clauses, compatible with the Prolog language. They allow the blend to be matched against specific composite concepts, situations, abstract ideas or similar cognitive contexts.

The supported frames are of three types, according to the elements (relations or concepts) matched in the blend and the existing elements present in one or more input spaces:

local frames compare the existence of elements between the blend space and the input spaces. When comparing relations, these are counted for a given label and must be connected to a concept (eg. count all the *isa,pw* relations connected to the *horse* concept) or in all the graph (count all the *isa,w* relations in the graph). The local frames ignore both edge transitivity (memoryless) and directionality, thus their naming. Examples of these frames are the Divago’s *aprojection/bprojection* frames, whose purpose is to compare the existence of a custom set of concepts in the blend; and Divago’s *aframe/bframe* which counts relations of a given mental space in blend. The frames nomenclature (*a* or *b*) is because each frame is related to a specific input space, eg. *a* to the *horse* and *b* to the *bird* input space.

pattern frames match the maximum number of concepts in the blend space, subject to the specified restrictions. With the usage of variables and atoms in a clause, the blender is able to detect a pattern of interconnected relations. The score for an individual pattern frame is 1 when the clause applies fully. Otherwise, Prolog iteratively chunks the frame clause’s predicates in individual clauses of one predicate and counts the number of predicates which are successfully solved. Then, the score will be $k/(n - 1)$, with k the number of successfully solved predicates and n the total amount of predicates of the frame. An example of a pattern frame is shown in Fig. 3.

delta frames are semantically equivalent to pattern frames with the difference that the instantiation of variables must be different between a mental space and the blend space. For instance, using the frame in Fig. 4 as a delta frame, the frame is maximised when the variables W, A, P are instantiated with different concepts from the blend space than those instantiated from the input space. The score for a delta frame is directly proportional to the amount of different instantiated concepts between the blend space and a mental space. When more than one instantiation for the variables is possible, the final score is the maximum of the individual scores. Our rationale is that in

$$\begin{aligned} & \text{ability}(A, \text{'bird/fly'}), \text{pw}(\text{'bird/wing'}, A), \\ & \text{pw}(\text{'horse/leg'}, A), \text{ability}(A, \text{'horse/run'}), \\ & \text{pw}(A2, A1), \text{pw}(A1, A), \text{purpose}(A2, \text{'horse/hear'}). \end{aligned}$$

Figure 3: Pattern frame of an A composed of a wing and a leg part, with the ability to fly and run, as well as a sub-part whose purpose is to hear.

$$\text{ability}(W, A), \text{purpose}(P, A), \text{pw}(P, W).$$

Figure 4: Delta frame *new_ability* of an entity W with the ability A and with a part P performing the purpose A .

this case there is at least one frame which maximises the difference between the blend space and a mental space and as such, there is a pattern which differs between a mental space and the blend by having the greatest amount of concepts. An example of a delta frame used in the experiments is shown in Fig. 4, which detects entities W with ability A and part P with purpose A . Hence, this frame gives importance to the emergence of entities with parts whose abilities and purposes are different than the ones present in the combined input space.

Blend chromosome

The chromosome of each blend is defined by a local analogy and a local blend space. Both analogy and blend space are stored in the chromosome and therefore specific to an individual blend. When the GA starts, the blend space of every chromosome is an empty space. Then, the analogy chunk of each chromosome is initialized as either a full copy or a random subset of the mappings contained in a random supplied analogy. The blender then selects a random subset of mappings from the blend’s analogy. For each mapping m_i , one of four operations is then executed: extract the first concept c_i ; extract the second concept c_j ; ignore the current mapping; or create a blend (fusion) c_k of both concepts. Depending on the chosen concept(s), a nearby set of relations touching the concept(s) are pulled from the input space into the blend space. The name of a blended concept c_k is the concatenation of both names separated by an underscore: $c_k = c_{i0} _ c_{i1}$. During each epoch, the GA applies a mutation to every blend in the population. Afterwards, all the blends are evaluated according to a fitness function.

Blend mutation

The mutation is applied to a blend in two steps: mutation of the blend mappings and of the blend space. The set of mappings is mutated as follows: setting them to fully match one of the supplied analogies; the random removal of one or more mappings from the blend’s analogy; and the random insertion of one (or more) mappings from the supplied analogies.

The mutation does not create mappings which do not exist in the supplied analogies. If this happened

there was no reason to require a list of analogies as the blender would wildly conceive random mappings. On the other hand, these mappings would likely be flawed, as the blender omits semantic and structural knowledge related to the concepts present in the input space. As such, our blender assumes there is an external adequate algorithm which supplies the analogies.

The mutation of the blend space is divided in five steps, transforming its structure as follows:

- addition of edges from the input space, including new concepts;
- removal of edges and/or concepts from the blend space;
- inclusion of two blended concepts and relations linked with one or both concepts. When two concepts a and b are blended together, the new concept is named $concept_a|concept_b$, eg. 'horse/leg|bird/leg';
- inclusion of one concept c_0 from a random mapping m_r with some (or all) relations associated with the other concept c_1 of the mapping $m_r = \{c_0, c_1\}$, replacing in those relations one concept by the other. Concepts c_0 and c_1 may be randomly swapped;
- selection of a random concept in the blend space, changing it and its associated relations to an opposing concept, according to a chosen random mapping.

For each epoch, the above steps are independently and randomly applied to each blend of the current population. The presence of the mutation operator is sufficient (Tate and Smith 1993) to allow the emergence of a diversity of blends through the evolutionary process.

How enlightened is that blend?

Divago used six optimality principles: Integration, Topology, Unpacking, Maximisation/Intensification of Vital Relations, Web and Relevance (Pereira 2005). The optimality principles are used to exert pressure towards stable, consistent and integrable blends.

Martins et al. (2016) did a deeper analysis on both the impact and importance of each individual principle in achieving a "good blend". The authors, supported by empirical experiments, suggest that five principles: Integration, Topology, Unpacking, Relevance, and Intensification / Maximisation of vital relations - could be enough for achieving interesting blends.

Encouraged by the study, we decided on walking new grounds with the idea of *Simplicity Theory* (ST) (Dessalles 2013) in mind, with the aim of studying simpler and intuitive methods in conceiving interesting blends. In ST, Dessalles asserts that the human mind is highly sensitive to any discrepancy in the complexity of information (Fig. 5). Motivated by the assumption that the human brain is sensitive to algorithmic complexity, the author alleges that the impact induced in people is proportional to how much simplicity is present in the information people are shown. This corresponds to the idea that much of cognition is regarding the compression or the elimination of redundancy (Chater and

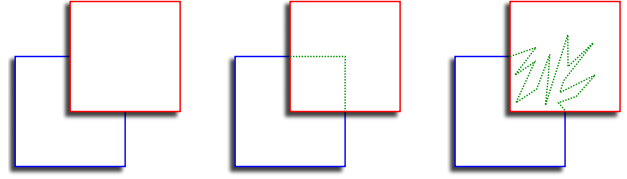


Figure 5: Simple and intricate interpretations for an occluded figure (blue). Best viewed in colour.

Vitányi 2003) in whatever information is the human brain processing. As entropy is related to the idea of information compression, our blender contains two forms of graph entropy as optimality measures. As such, the blender currently neglects most optimality principles of CB theory, accepting the fact that we may lose some consistency and coherence in the obtained blends. However, with the execution of experiments we expect conclusions to be made regarding a future study regarding a trade-off of some (or all) of theory's principles, perhaps including the study of new measures. Nevertheless, our blender requires other heuristics which either individually either combined, guide the blending process towards the direction of "good" blends.

The assessment of the blends is done in four perspectives: topology, entropy, frame related and general informative measures. These are explained below:

Topology

We follow the definition of Topology used in Divago, where topology exerts a form of inertia in the blending process. For any mental space and any element in that space projected into the blend, it is optimal for the relations of the element in the blend to match the relations of its counterpart. A relation present in the blend space is topologically correct when it occurs in at least one of the mental spaces. The topology measure is defined as a ratio of topologically correct relations present in the blend space. The definition we used for topology is the same as defined in (Pereira 2005), section 4.2. For reference, it is given as follows:

Topology: for a set $TC \subseteq CM_b$ of *topologically correct* relations, defined as

$$TC = \{r(x, y) : r(x, y) \in CM_1 \cup \dots \cup CM_n\}, \quad (1)$$

where $CM_1 \dots CM_n$ correspond to the concept maps of the n input/mental spaces. The topology measure is then calculated as the ratio:

$$Topology = \frac{\#TC}{\#CM_b}. \quad (2)$$

Topology drives against change in the blend space because while preserving a similar topological configuration as the input space. According to its definition, the blend should preserve the same neighbourhood relations between every concept in the blend space.

Entropy

Various measures of graph entropy exist (Dehmer and Mowshowitz 2011). To measure one form of complexity (or compressibility) of the blend space, we implemented two entropy measures based on Shannon’s entropy. These are calculated according to the relation labels of the blend space. The majority of the blend’s concepts are unique and in turn, we decided to disregard entropy related to the concepts.

BlendVille contains two entropy measures for the blend’s relations: 0-order and 1-order. The difference is due to the relevance given or not to sequences and the directivity of relations. We explain these below.

0-order entropy: this measure of entropy coincides with Shannon’s classical entropy, defined as follows: given a discrete random variable R with n symbols r_0, \dots, r_n and probability density function $P(R)$ defined for each symbol r_i , the 0-order entropy $H(R)$ is:

$$H(R) = - \sum_{i=1}^n P(r_i) \log_e P(r_i). \quad (3)$$

The random variable R corresponds to the set of all the relations (ie. $R = \{isa, pw, ability, \dots\}$) present in the blend space. Accordingly, $P(r_i)$ is the relative frequency of the relation r_i in the blend space. Hence, the probability density function $P(R_i)$ of a relation r_i is defined as the ratio between that relation’s label absolute frequency and all the relations present in the blend graph:

$$P(r_i) = \frac{F_{r_i}}{\sum_{j=1}^n F_{r_j}}. \quad (4)$$

The reason for this measure being named 0-order is because it corresponds to a stateless description of the blend space, having no interpretation on the sequences of relations. This measure applied to a fully connected or disconnected graph would naturally result in the same measured value. However, it allows one assessment of the redundancy or uniqueness of labels of the relations in the blend.

1-order entropy: This measure is defined as an extension to the 0-order entropy, with the difference that it takes into account pairs of consecutive relations $\{r_{i0}, r_{i1}\}$ and their directivity (Fig. 6). For instance, the relations connected (through the horse concept) $isa(horse, animal)$ and $pw(horse, leg)$ generate the relation pair $\{isa, pw\}$. The calculus of the 1-order entropy is done analogous to the steps above, with summations adapted to include pairs of connected relations and their directivity, instead of single relations. The directivity of the pairs is defined as related to a common concept: if in the same direction; incoming; or outgoing to the concept (Fig. 6).

Frame evaluation

To have a meaningful interpretation and purpose, BlendVille uses frames to define the content and context

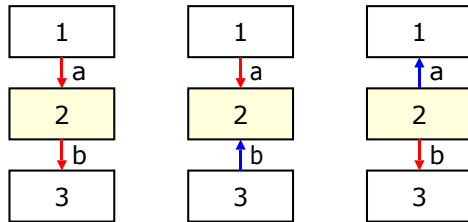


Figure 6: 1-order entropy patterns for the pairs of relations $\{a, b\}$: in the same direction, incoming or outgoing to the common concept “2”. Best viewed in colour.

of the blend, conforming to CB’s view of mental spaces. The types of frames exposed in the previous section *Frames* are evaluated and given individual scores according to their type: *concept frames* (for a given mental space), *edge frames* (also for a given mental space), *delta frames* and *pattern frames*.

In the case of multiple delta Frames and/or pattern Frames, our blender does not currently prioritise one delta or pattern frames over another. We expect this to be improved in the future. In the situations where there are multiple delta/pattern frames to evaluate, the system gives a score to a frame proportional to the matching of each individual frame’s predicates (range $[0 \dots 1[$ or 1 if the frames matches fully) and proportional to the number of applicable frames.

General Informative

The last two measures are not related to semantics but are used to fine-tune the global structure of the blend graph and the relative contribution of the input spaces. These are the number of *graph islands* and the amount of *inter-space edges*.

graph islands counts what in graph theory is defined as the number of connected components (islands) in the blend space. It is calculated in linear time using breadth first search for all the concepts in the graph.

inter-space edges is the number of relations present in the blend space which connect concepts of distinct mental spaces. An example is the relation $pw('bird/wing', 'horse/horse')$. The exception is when a blended concept of different mental spaces, such as $'horse/leg/bird/leg'$ is present in the relation. In this case, as the relation associates concepts of the same mental space as subject and object, we consider the relation as not inter-space.

Novelty and Usefulness

For assessing the quality of the generated blends and validating our blender against Divago, we used the same measures *novelty* and *usefulness* as defined by Pereira, themselves based on the work by Ritchie (2007). Ritchie considers *typicality* and *value* where Pereira defines *novelty* as the opposite of *typicality* and *usefulness* as a synonym of *value* (Pereira 2005). Both measures are defined next:

novelty describes a measurement of non typicality, surprise, a change in information. Novelty of a blend, which Pereira describes as the “*converse of Ritchie’s typicality function*”, is defined as a function of $d(b, x)$ and the size of the blend $size_b$.

Let x be one of the n input spaces and b the blend space. Then, $d(b, x)$ is “*the sum of the relations that: 1) belong to b and that are missing in x with 2) those that belong to x and are missing in b* ”. Next, an edit distance $distance(b)$ is defined as:

$$distance(b) = \frac{\min(d(b, x_1), \dots, d(b, x_n))}{size_b}. \quad (5)$$

The measure novelty of the blend b is calculated as:

$$novelty(b) = \begin{cases} 1 & distance(b) > 1 \\ distance(b) & \text{otherwise} \end{cases}. \quad (6)$$

usefulness evaluates the blend according to a purpose. In Divago, the purpose was defined before an experiment, after which the usefulness of the blend is assessed. Furthermore, the purpose is defined as the blend having an exact similarity (both structural and semantic) to a specific conceptual map (semantic network).

Given a conceptual map t and a blend space b , $d(b, t)$ is the sum of relations that belong to t and are missing in the blend b . Then, usefulness is:

$$usefulness(b) = 1 - \frac{d(b, t)}{size_b}. \quad (7)$$

Next, we experimented our blender in various situations and assessed its results with the above measures. These experiments are exposed in the next section.

Experiments and discussion

The experiments were carried out with the conceptual maps *horse* and *bird*, available in Pereira’s PhD thesis (Pereira 2005) in Tables 5.1 and 5.2 of page 100. The three analogies (sets of mappings) for the *horse* and *bird* experiment are also found in the same thesis, in Fig. 4.5, page 110. The *delta frame* used for detecting a new ability is given in Fig. 4. For the usefulness, the target blend was set as a conceptual map representing the *Pegasus*. The Pegasus is defined as containing the same conceptual map of the *horse* to which *two wings* were added as well as the *ability* to *fly*. Therefore, the five following relations were added:

`ability('horse/horse', 'bird/fly'),`
`pw('bird/wing', 'horse/horse'),`
`quantity('bird/wing', '2'),`
`purpose('bird/wing', 'bird/fly'),`
`motion_process('horse/horse', 'bird/fly').`

Testing the measures

The measures were evaluated either individually or combined. Their impact and validity in novelty and usefulness are described next. Unless otherwise noted, all

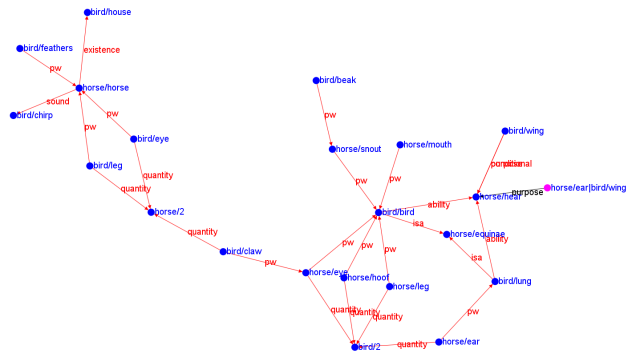


Figure 7: Example of a blend with low 1-order entropy. Edges in red are inter-space relations. Best viewed in colour.

experiments minimised the measure *number of graphs islands* (components) in order to obtain a fully inter-connected blend space.

Topology When we used exclusively this measure as the fitness function, the blend ended as a projected copy of the input space. Although one could expect novelty to be inversely correlated with topology, the novelty measure - as defined by Pereira - is specific to the exact structure (semantic and relational) of the input spaces. On the other hand, topology is defined on the relations directly connected to each concept of the blend space. Therefore it does not guarantee topology at a higher structural level. We observed that when the blend’s topology had in average score of 100%, novelty was contained in the interval of [95%, 100%]. This happened as a result of the stochastic nature of the GA, which allowed the blends to randomly evolve in both the blend space and the set of mappings. On all the topology experiments there was at least a dozen of inter-space relations connecting concepts of different mental spaces. By itself, this is enough to maximise novelty.

On the other hand, usefulness was between 25% and 50%, which somewhat demonstrates that on its own, there is no relation between topology and usefulness. This is expected, as the above definition of topology has no reason to justify the emergence of a *Pegasus* blend, even though the *Pegasus* mental space is defined as the union of the *horse* mental space with five specific relations of the *Pegasus*, with the particularity that 90% of *Pegasus’* relations are from the *horse* input space. This demonstrates that topology reflects a statistical description of the relations labels and ignores the global structure of both the blend and input spaces.

Entropy

These measures had no impact in the semantic structure of the blend space. However, they did have a definitive influence in the compression of the blend, as well as an effect in the presence of redundant structures.

- **0-order entropy** directly affects the variety of relations of each type in the blend. This allows two limits:

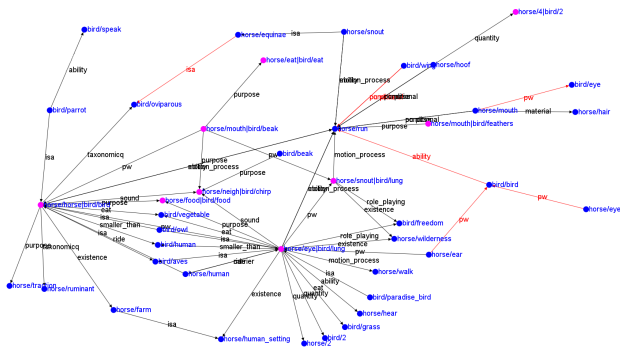


Figure 8: A blend with high 1-order entropy. Edges in red are inter-space relations. Best viewed in colour.

when 0-order entropy is maximum the blend has the highest amount of unique relations of a certain label ($1 \times isa$, $1 \times pw$, $1 \times ability$, etc.) and the opposing case, when entropy is minimum the blend has only relations of one label, ie. $1 \times isa$ or $10^{99} \times isa$. However, 0-order entropy does not affect the total amount of relations present in the graph, but in fact the relative amount of relations of each type. Given a blend with 2 *isa* and 5 *pw* relations, its 0-order entropy is the same as another blend with 20 *isa* and 50 *pw* relations, or in another words, the blend is allowed to contain n groups of $\{2 \times isa \cup 5 \times pw\}$ relations without affecting the value of the measure. This is because 0-order entropy is defined on the *relative* probabilities $P(X)$ of the discrete variable X (the labels of the relations) occurring in the blend.

- **1-order entropy** - This measure affects the variety of sequences of relations present in the blend space. For instance, if a certain blend has the relation pattern *pw*(A,B) and *quantity*(A,C), this pattern is allowed to materialize elsewhere in the blend space without affecting 1-order entropy (Fig. 7). Therefore, this usage of this measure allows the appearance of repeating structures in the blend space or, on the other hand, the manifestation of diversified structures in the blend (Fig. 8).

An understandable observation is that the entropy measures have no obvious correlation with novelty, much less with usefulness. Even when the blend's entropy is equal to one or more of the input spaces (or the conceptual map of the Pegasus), that equality would not imply a structural and semantic similarity between those spaces. This is expected as entropy is a statistical description of information.

Frames

The local frames (a/bprojection) bring the concepts of their input space into the blend space. Similarly, the a/bframes allow the blend space to have a statistical distribution of relations equal to their specific input space. Thus, achieving a useful blend in the context of the Pegasus concept map requires the *aprojection*

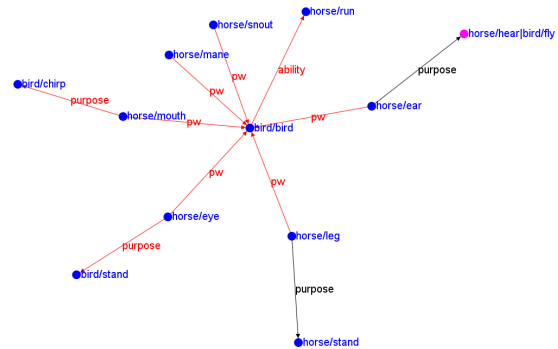


Figure 9: Example of a blend defined by the pattern frame in Fig. 3. Best viewed in colour.

and *aframe* frames to be present in the blend, in order to emerge the structure of the horse mental space in the blend space. The inclusion of the delta frame *new_ability* in Fig. 4 allowed the blend to differ from the input spaces, materialising a new ability in the blend space which did not exist in any of the input spaces, increasing the blend's novelty.

The inclusion of the horse concepts (aprojection), horse relations (aframe), delta frame *new_ability* and topology enabled the creation of blends with a usefulness of 93%...95%. However, without more elaborated forms of expressing the frames required to represent the Pegasus mental space, it is not possible with our current blender to obtain the Pegasus conceptual map. This also exposes what we consider is an issue with the definition of usefulness.

Adding pattern frames to the fitness function favoured the manifestation of blends with more elaborated semantic interpretations. Using the pattern frame in Fig. 3, the delta frame *new_ability*, maximising the number of inter-space relations while minimising the 1-order entropy allowed the blender to generate interesting blends, one of such blends is shown in Fig. 9. That blend shows is a bird with a mouth of a horse, able to chirp, eyes used to stand up and with a horse mantle. Intriguing to note that the ears are used to fly instead of the wings, being these used to run. In separate experiments we witnessed blends that represented animals which could hear using their wings, being these attached to their snout. Kind of giving a new purpose to the definition of wing... in the form of a tympanic membrane.

Graph Islands

This measure is useful in creating strongly connected blends. It allows the penalisation of blends whose spaces have disconnected components, ie., with no detached relations “floating” around.

Inter-Space edges

Reinforcing the number of inter-space relations in the blend space tends to maximise novelty, as the blend

space becomes filled with a mixture of connected concepts from different input spaces. In the horse-bird experiment which has two input spaces, the blend space ends with roughly 50% of the concepts of each input space. As novelty measures the amount of missing relations (and related concepts) from each input space, the emergence of relations connecting concepts from different input spaces naturally increases novelty.

Comments on Novelty and Usefulness

We can confirm that *novelty* as defined in Divago does indeed measure a modification of semantic structures in the blend, when compared to the input spaces. However, it only measures a total mismatch of a relation, not having into account, for example, when in a relation the only change from an input space to the blend space is the modification of a single concept, or of the label of the relation itself. We believe novelty should be proportional to the minimal possible change in the representation of the blend space.

We agree with Ritchie (2007) regarding usefulness. It should not be defined strictly according to a purpose, but in a more general and fuzzier perspective. This is the main reason why during the pegasus experiment our blender was not able to reach a score of 100% in usefulness. We believe that the change in usefulness should correspond to Ritchie's definition of *value*, which rates the worth or importance of the newly created artefact, not necessarily before its creation.

Future Work

We expect a great deal of work to be done. We envision the improvement of the measures *novelty* and *usefulness*. Regarding the entropy we expect at least two developments: multiple orders of entropy which will allow the emergence of structural redundancy at various levels; and the involvement of both semantics and frames in the calculation of entropy. The exploration of different types of frames, as well as the latest developments in image schemas, is also expected to be pursued.

Conclusion

In this work we proposed an evolutionary system modelled on CB theory. The blender implementation - BlendVille - receives input spaces, frames, analogies and outputs blends capable of displaying novelty. We feel that the system exhibits a form of creativity. Our system has its roots on Divago but follows a different recipe regarding the use of optimality principles in order to generate understandable, independent and coherent artefacts. The main ingredient is based on information theory, mainly the concept of entropy. Therefore, we think our system allows the emergence of a form of redundancy in the generated blends, in accordance with the idea present in Simplicity Theory.

Acknowledgements

This research was funded through EC funding for the project ConCreTe (grant number 611733) that acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission.

References

- Chater, N., and Vitányi, P. 2003. Simplicity: a unifying principle in cognitive science? *Trends in cognitive sciences* 7(1):19–22.
- Dehmer, M., and Mowshowitz, A. 2011. A history of graph entropy measures. *Information Sciences* 181(1):57–78.
- Dessalles, J.-L. 2013. Algorithmic simplicity and relevance. In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*. Springer. 119–130.
- Fauconnier, G., and Turner, M. 2002. *The Way We Think*. New York: Basic Books.
- Martins, P.; Pollak, S.; Urbančič, T.; and Cardoso, A. 2016. Optimality principles in computational approaches to conceptual blending: Do we need them (at) all? In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016)*. Paris, France: Sony CSL.
- Pereira, F. C. 2005. *Creativity and AI: A Conceptual Blending approach*. Ph.D. Dissertation, University of Coimbra.
- Ritchie, G. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17(1):67–99.
- Schorlemmer, M.; Smaill, A.; Kühnberger, K.-U.; Kutz, O.; Colton, S.; Cambouropoulos, E.; and Pease, A. 2014. Coinvent: Towards a computational concept invention theory. In *Proceedings of the 5th Int. Conference on Computational Creativity, ICC-14, Ljubljana, Slovenia*.
- Tate, D. M., and Smith, A. E. 1993. Expected allele coverage and the role of mutation in genetic algorithms. In *ICGA*, 31–37.
- Veale, T., and Keane, M. 1997. The competence of sub-optimal structure mapping on hard analogies. In *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI-97.
- Žnidaršič, M.; Cardoso, A.; Gervás, P.; Martins, P.; Hervás, R.; Alves, A. O.; Oliveira, H.; Xiao, P.; Linkola, S.; Toivonen, H.; Kranjc, J.; and Lavrač, N. 2016. Computational creativity infrastructure for online software composition: A conceptual blending use case. In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016)*. Paris, France: Sony CSL.

Encouraging p-creative behaviour with computational curiosity

Kazjon Grace^{1, 2}, Mary Lou Maher² Maryam Mohseni², and Rafael Pérez y Pérez³

¹The University of Sydney, ²UNC Charlotte, and ³ UAM Cuajimalpa

¹Australia, ²USA, and ³ Mexico

kazjon.grace@sydney.edu.au, {m.maher, mmohseni}@unc.edu, rperez@correo.cua.uam.mx

Abstract

A concept, design or other artefact is p-creative when it is simultaneously novel and valuable for a specific individual. This is defined by contrast to h-creative artefacts, which are novel and valuable for a society as a whole. When we talk about p-creativity in computational systems we usually mean that something is creative to the system itself: the system has its own experiences and goals, and with them judges novelty and value. We propose an alternative approach aimed at *simulating* what a specific human user will find p-creative in order to *stimulate* that user towards p-creative behaviour. We define a framework for simulating curiosity, explore several domains in which it could be applied, and describe some preliminary results from a system designed to suggest papers for students to read that they would find surprising. We end the paper with a discussion of how this model can be extended to generate *framing narratives* that combine content from different artefacts that encourages p-creative behaviour.

Introduction

An artefact observed by a creative agent (human or artificial) is p-creative (“psychologically” creative) when that agent considers it novel and valuable (Boden 2004), regardless of whether other agents or the society as a whole would agree. Increasing the number of such observations is clearly beneficial for creativity, but we argue it may also be of benefit outside creative domains. The more different kinds of food we eat, the healthier we tend to be (Vadiveloo et al. 2015). The more broadly we read fiction, the greater our ability to understand others’ emotions (Kidd and Castano 2013). Employees with both breadth and depth – “t-shaped” people – are sought out for their collaboration skills (Berger 2010). In each of these situations it is *seeking p-creative experiences*, rather than any specific goal, that is desirable. This paper describes a framework for interactive systems that encourage p-creative behaviour in their users, with applications both within and outside creativity.

Encouraging p-creative behaviour in a user is distinct from encouraging creative behaviour in general. The majority of computationally co-creative systems drive the user towards their best estimate of h-creativity (or “historical” creativity – artefacts judged to be creative by society as whole). Encouraging a user to pursue p-creativity requires knowing

what *they* will find novel and valuable. Our reasoning for trying to directly encourage p-creativity is based on its impact on the user’s motivations. Unexpected discoveries play an important role in driving the user towards more creative outcomes (Suwa, Gero, and Purcell 2000), a partial explanation for which may be the impact of curiosity on learning and performance (Reio and Wiswell 2000). Our goal is to develop creative systems capable of simulating the user’s novelty and value functions with sufficient fidelity to suggest p-creative actions. We hypothesise that these systems may act as a kind of *curiosity engine*, repeatedly stimulating curiosity and encouraging behavioural diversification.

This paper is structured in three parts. We first describe a framework for this kind of system which we call the *Personalised Curiosity Engine*, or PQE (pronounced “pique”). We then describe an initial prototype of one component of the PQE system in the domain of text: a model of what makes a document unexpected. We conclude with a discussion of how the task of suggesting content in PQE systems could be framed as a form of narrative generation.

PQE systems persuade their users to take p-creative actions by *simulating* their novelty and value functions in order to *stimulate* their curiosity. They could be applied to creative tasks as a way to overcome fixation (Purcell and Gero 1996) and encourage diversity. They could also be applied to any other situation in which diverse action is of benefit, such as food or reading. PQE systems applied outside of traditionally “creative” domains are a kind of *persuasive computing* (Fogg 2009). These systems draw on computational creativity techniques to inspire curiosity and persuade users to consider more diverse actions.

p-Creative experiences are motivating

Behaviour change systems are a class of persuasive interactive system concerned with encouraging users to make sustainable changes to their behaviour in domains like education and health (Fogg 2002). Despite some early successes, behaviour change has remained largely an unsolved problem, for the simple reason that old habits die hard (Klasnja, Consolvo, and Pratt 2011). Using extrinsic rewards or social reinforcement to motivate significant sustained behaviour change is extraordinarily challenging, regardless of whether technology is involved. Our approach leverages a different motivator: curiosity.

An individual’s motivation to perform an activity can belong to two broad classes: the extrinsic motivation to perform an activity for a reward (e.g. money, status or grades), and the intrinsic motivation to perform an activity for its own sake. Intrinsic motivation leads to greater interest, confidence, and performance (Ryan and Deci 2000).

Intrinsic motivation is tightly connected to the drive for self determination and self growth, and represents a principal source of enjoyment throughout life (Csikszentmihalyi and Rathunde 1992). Curiosity is one model for intrinsic motivation: it is the drive to seek novel stimuli for the sake of learning alone (Merrick and Maher 2009; Barto, Singh, and Chentanez 2004). The state of curiosity (as distinct from the trait of curiosity, see (Berlyne 1966)) has been modeled as seeking stimuli that are new enough to arouse interest but not so new as to cause disgust (Saunders and Gero 2001), and as seeking stimuli that most improve one’s model of the world (Schmidhuber 2010).

Our approach leverages these cognitive models to simulate the curiosity of an individual and suggest recipes that will best stimulate their intrinsic motivational drive. Past computational models of curiosity have been used to generate novel content (Saunders and Gero 2001; Merrick and Maher 2009). Our approach differs in that we will simulate what an individual user will find curious, rather than imbue curiosity within an intelligent interactive system.

PQE: The Personalised Curiosity Engine

The PQE framework is a high-level approach to building systems that encourage p-creative behaviour, both in creative domains and everyday life. The underlying assumption is that a human user is engaged in taking actions within a particular domain with the goal of having p-creative experiences. The actions a user takes will always include engaging with artefacts within the domain, but may also include creating new ones. For example, in a music domain the user can listen to tracks composed by others, and may or may not be engaged in composition herself. In a cooking domain the user will eat and/or cook recipes written by others, and may or may not create her own. Describing interactions with our framework this abstractly lets us also apply it in cases where the user is not creating new artefacts, such as helping a graduate student discover new research papers and topics. This parallels the notion of a “serendipitous” recommendation (Herlocker et al. 2004), a notion which has also been studied in creative contexts (Corneli et al. 2014), although we avoid that term as it evokes the notion of such discoveries occurring by chance. Our long-term hypothesis is that repeatedly engaging with p-creative stimuli suggested by a PQE system would, over time, diversify a user’s preferences.

Simulating p-creative evaluation in PQE

To make p-creative suggestions to our user (i.e. *stimulate* them towards p-creative behaviour) we must first estimate what they will find p-creative (i.e. *simulate* their evaluation of p-creativity). Standard models of novelty are built from the system’s knowledge, with the assumption that it reflects the domain as a whole. In our case we must first develop a novelty model based on the knowledge of the user.

Techniques for approximating the knowledge (for novelty modelling) and preferences (for value modelling) of an individual fall under the area of *personalisation and user modelling* (Kay 1994). Modelling the value function of an individual (their “p-value”, by analogy to p-creativity) requires knowing that individual’s preferences within the domain.

Perhaps the most common paradigm for modelling preferences is the recommender system (Ricci, Rokach, and Shapira 2011). Recommenders use a model of the user and/or the items to select a set of results likely to be chosen by the user. The goal of these approaches is to use the preference ratings of a small set of known artefacts to estimate preferences across the whole domain. Very efficient algorithms exist for both user-based (“collaborative”) and item-based (“content-based”) preference modelling, and systems implementing those models are near-universal in online commerce. These preference models present a natural starting point for simulating users’ value functions.

Modelling individual novelty (i.e. “p-novelty”) is more complex. Our previous work has developed models of novelty based on expectations (Grace and Maher 2014; Grace et al. 2015). We define a novel artefact as one that violates an observer’s expectations, where those expectations are based on the observer’s past experiences (Baldi and Itti 2010). This makes novelty fundamentally subjective: to simulate the novelty of an individual one must estimate their knowledge of the domain. We propose an approach to simulating a user’s novelty that does not require a full record of what domain artefacts they have observed.

Stimulating p-creative behaviour in PQE

We propose systems that encourage p-creative behaviour by making suggestions for what actions to take. The simplest approach to delivering suggestions to the users is to wait for the user to query the PQE system and then provide a list of suggestions as a kind of search result. This is the approach adopted by recommender systems: the user is known to be looking for something, so tailor the information that is retrieved based on the available user model. While simple to implement, this approach may not effectively stimulate p-creative behaviour. Firstly, users may not be in a mindset compatible with the “search” metaphor, as they may be actively creating or acting independently.

Suggestions, like creative artefacts, might reasonably be expected to be more appreciated when explained (Moran and Carroll 1996). This draws to mind the concept of creative systems that provide *framing* for their generative acts (Charnley, Pease, and Colton 2012). One possibility for going beyond simple recommendation is to provide compelling framing alongside the suggestions made by our systems, an approach we discuss later in this paper.

Structure of the PQE framework

The structure of the framework can be seen in Figure 1. PQE Users provide feedback on their actions in the domain in the form of preferences and what surprises them. This feedback is used to estimate their value and novelty functions respectively. This feedback may be prompted in response to observing a specific new artefact, or it may come in the form

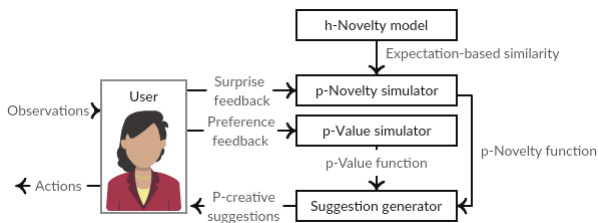


Figure 1: The three processes of the domain-general PQE framework and how they interact with a user to encourage p-creative behaviour.

of more general questions about likes, dislikes, familiarities and surprises. This feedback is then used by the *novelty simulator* and *value simulator* processes.

The **value simulator**'s task is to estimate the user's value function for the whole domain, given information about some subset of that domain for which feedback has been provided. This feedback comes in the form of preferences or ratings: value judgements of artefacts, attributes of artefacts, or classes to which artefacts belong. The value simulator process must infer the user's values across the whole domain (their personal "value function") from this feedback. It may draw on knowledge about the relationships between artefacts and/or users to do so.

The **novelty simulator**'s task parallels that of the value simulator: it must estimate the user's novelty function for the whole domain feedback. Like the user's value feedback, this comes in the form of judgements about novelty. The user reports that they find some artefacts or features to be low-novelty ("unsurprising") and others high-novelty ("surprising"). Unlike the value simulator this sparse feedback is insufficient to infer a novelty function for the domain. The h-novelty model provides the required additional knowledge.

The **h-novelty model** is based on our prior work in expectation-based models of novelty. These models take the entire database and construct a set of expectations in the form of conditional probabilities. PQE uses this model to provide a similarity metric that the novelty simulator can use to compare objects that the user has rated to objects that they have not. This similarity metric treats objects that are based on similar expectations as being similar. This results in estimates of how surprising a user will find a new artefact that are based on how surprising they have found similar combinations of surprising features elsewhere. For example, a user who found a mix of sweet and sour flavours totally unsurprising in a stir fry recipe will likely find the same combination unsurprising in a salad. This then allows the novelty simulator to operate in the same way that the value simulator does: using known ratings to infer unknown ones.

The **suggestion generator** uses the inferred value and novelty functions to provide recommendations to the user. These are intended to influence what areas of the domain the user explores (i.e. what artefacts they observe, create, consume or otherwise engage with). We do not specify the exact form these suggestions may take, but examples include

search results or a single recommended action.

The PQE framework is applicable to computational co-creativity contexts (in which a human's creative acts are being supported or enhanced), but also to other domains that are not traditionally considered "creative". Discovering new things that are simultaneously novel and valuable to you can happen in any domain, this is a core component of the notion of *everyday creativity* (Runco and Richards 1997). In the following section we describe a pilot implementation of one component of the framework, the h-novelty model, in the domain of research papers.

Applying PQE to Research Papers

We are prototyping PQE in the domain of research papers, as part of an eventual future system that encourages students to read more broadly. Research papers are an example of a creative domain where unexpected discoveries are valuable and may lead to future creative research acts that generate their own papers. Our current goal is to understand what makes a research paper unexpected and to build an h-novelty model for this domain. The implementation presented here does not yet include personalisation.

In this paper we describe one promising model of research paper novelty, and demonstrate some preliminary results in the form of highly novel and highly not novel papers. Our next step will be to validate these results against the novelty judgements of humans who are experts in the research field of the papers in our corpus. If the experts are in agreement, then they can be expected to correlate with h-creativity in this domain (Amabile 1996).

For the purposes of these experiments we use a database of 298,000 research paper abstracts from the ACM Digital Library¹. These documents span many of the most prominent journals, magazines and conferences in computing research. Abstracts were used as a summary of each paper.

Modelling expectations in text

For this prototype we defined novelty as exhibiting unexpected conceptual combinations, a kind of *relational expectation* (Grace et al. 2015). In this section we explain the details of the process by which we identify concepts within text documents, compute their relationships, and then assess the unexpectedness of the combinations that appear in abstracts of the published papers.

We adopt a topic modelling approach to inducing concepts from our corpus (Blei, Ng, and Jordan 2003). A topic model is a probabilistic graphical model (a type of statistical model) for learning the themes that occur in a collection of documents. First each document is represented as a "bag of words", an approach that ignores word order and context in order to provide a unified vector representation for each document. These models then produce a set of "topics", each consisting of a distribution over all the words in the corpus. This is based on the modelling assumption that topics are a probabilistic mixture of all the words in the corpus. Words which feature strongly in a topic are assigned a relatively high probability. For example, our model produced

¹<http://dl.acm.org>

a topic which assigned the greatest probability mass to the words “network”, “node”, “protocol”, “communicat+”², and “rout+”. This topic clearly relates to computer networking.

Documents are then assigned different proportions of each topic, as in a mixture model. For example a paper on the Internet of Things may be drawn from 60% the above networking topic and 40% from a combination of topics about sensing, the Internet and experimentation. Topics are not labelled by the system and they are not guaranteed to be comprised of a single theme that is easily human-comprehensible, but they are usually at least moderately interpretable as in the example above.

We base our model of expectations on an extension of the basic topic modelling algorithm called a Correlated Topic Model, or CTM (Blei and Lafferty 2007). The advantage of this specific algorithm is that it allows for topics to be more or less correlated, i.e. the networking topic described above occurs more frequently with a topic about cybersecurity than it does with a topic about image recognition, as those themes are more conceptually related. This forms the basis of our expectation model: topics are concepts inferred from the dataset, and the correlations between topics give us a basis for what combinations of concepts are unexpected. Our prototype uses the R package “STM” (Roberts, Stewart, and Tingley 2014) to construct these models (STM is another topic model extension that is equivalent to a fast CTM implementation in some configurations). We use the default number of topics, 40, in our investigation.

In our previous models of relational expectation we have argued that the overall novelty of an artefact should be equal to the most novel concept or combination of concepts within that artefact (Grace et al. 2015). As a thought experiment on why we prefer this approach over averaging or combining multiple surprising artefact components, consider two fish. The first fish is slightly longer than expected, a slightly unexpected shade of blue, and has slightly bigger eyes than you would normally see. The second fish is physically unremarkable but can sing like a classically-trained soprano.

Our model of expectation bases the novelty of a text as the lowest (i.e. highest negative) correlation coefficient among all pairs of topics significantly present in that text, and the proportion of the document which contains that pair. We determine whether a topic is “significantly present” in a document using a topic proportion threshold of 0.1 (i.e. the document is at least 10% comprised of that topic). The formula can be seen in Equation 1, given a document $d = [t_i, t_j, \dots, t_n]$ consisting of the set of topics significantly present. t_i and t_j are pair of topics in d which have the lowest (i.e. highest negative) correlation coefficient.

$$\left(\text{CovMat}_{t_i, t_j} / \min_{k=1 \dots K, l=1 \dots K} (\text{CovMat}_{k, l}) \right) * \frac{1}{2(\min(\text{prop}(d, t_i), \text{prop}(d, t_j)))} \quad (1)$$

²We use “+” to denote a stem formed from the combination of multiple words with the same root, e.g. “communicate”, “communication” and “communicator”. Singular word forms are always combined with their plurals and are not marked.

Where CovMat is the covariance matrix for the topic model and $\text{prop}(d, t)$ is a function that returns the proportion of a document d that is comprised of a topic t . The first term is the novelty of the document’s most novel topic combination, expressed as a proportion of the most novel topic combination in the model. The second term is twice the smaller of the smaller of the two proportions of the document that come from the novel topics. The product of the two gives the normalised unexpectedness of the most unexpected topic combination weighted by how much of the document is made up of that combination.

The second term of the novelty equation was originally a sum of the two topic proportions, but we found this to favour documents that just passed the significance threshold with one topic, and were thus not particularly surprising. We adopted the minimum of the two topic proportions to weight our novelty measure towards documents that contained substantial amounts of both unexpected topics.

A document composed of 50% of each the two most novel topics in the model would be given a novelty of 1. A document containing at-most independent topics would have a novelty of 0. Documents containing only positively correlated topics have negative novelty. Documents containing a lot of a moderately novel topic combination will be rated more novel than documents containing only a little of the most novel pair of topics.

Results

The top five words of each of the 20 topics in our CTM trained on the ACM Digital Library’s abstracts were:

1. *image+, method, object, use, surfac+*
2. *application, service, mobile, provide+, resourc+*
3. *model, simulat+, use, process, operat+*
4. *comput+, will, student, learn, course*
5. *program, languag, code, use, implement+*
6. *system, design, develop, softwar, tool*
7. *perform, memor+, parallel, execut+, processor*
8. *search, propos+, method, feature, result*
9. *network, node, protocol, sensor, rout+*
10. *user, inform+, web, content, use*
11. *data, quer+, database, efficien+, large*
12. *algorithm, problem, graph, comput+, time*
13. *framework, structur+, specif+, approach, relat+*
14. *method, test, measur+, predict, use*
15. *result, analys+, evaluat+, stud+, effect*
16. *problem, strateg+, agent, decision, mechan+*
17. *power, design, energ+, circuit, propos+*
18. *interact, user, use, interface, game*
19. *secur+, attack, policy, privacy, can*
20. *research, social, stud+, group, communit+*

The majority of these topics have clear meanings within the domain of computer science research. For example, topic 2 is clearly about mobile services and their associated infrastructure, while topic 5 is clearly about programming languages and their features. Topics 6, 8, 13, 14, and 15 relate to the language used to describe research itself, rather than specific sub-fields of content. Topics 4, 7, 17, 18 and 19

clearly refer to specific research topics, respectfully computing education, parallel computing, chip engineering, human-computer interaction, and cybersecurity. We have not performed any formal verification of the validity of this model, but they pass casual inspection as reasonably reflective of the abstracts from the ACM digital library.

Using this model we present three of the most unexpected and three of the most expected papers from our dataset of abstracts. In each case we provide a link that can be used to view the abstracts at the ACM Digital Library. The three highly surprising documents can be seen in Table 1

The first paper in Table 1 combines Topic 2 (mobile service infrastructure) with a Topic 12 (algorithms research). While to a casual observer these seem highly related topics, they are actually fairly contra-indicated within computing research. Mobile services are a highly applied topic, while algorithms are basic research. The abstract for this paper is also highly unusual, containing mathematical formalisms to describe the problem the paper solves. The topic model picked up on this abstract's unusual wording.

The second paper in Table 1 has an extremely short abstract, which appears to have resulted in the topic model assigning very concentrated topic proportions to it. This abstract belongs to a magazine article, a format which does not traditionally have article abstracts – closer inspection reveals the abstract to be a pull quote from the first page. This, other than highlighting the challenge of obtaining quality data even from one of the most reputable archives of computing research, shows the disadvantage of working with abstracts. Neither of these papers is particularly novel-seeming, but their abstracts are certainly highly atypical.

The third paper is the most traditionally “novel” according to our topic model approach. The paper combines topic 20, which appears to be associated with social science in computing, with topic 12, the algorithms and graph research topic. This is because the paper describes algorithms for finding particular kinds of groups of people in social graphs. We also note that topic 12 was present in all of the top three papers, and actually the top four most surprising topic combinations in the dataset. The algorithms research theme appears to be the most polarising of all twenty topics: it is highly correlated and highly uncorrelated with many other topics, and independent from few.

By contrast all three of the papers in Table 2 are highly concentrated combinations of two most related topics in our model: 13 and 5. Topic 5 is about programming languages, while topic 13 is about the more ephemeral artefacts of computing research: frameworks, approaches, models, and other structures. In all three papers the abstract describes a new structure (two frameworks and an approach) that contributes to programming languages research in some way. This – for better or worse – appears to be by far the least surprising kind of paper in the ACM DL.

Our current prototype only consists of the h-novelty model process in the PQE framework (see Figure 1). We are developing a recommender systems approach to answering user searches with a list of results that is simultaneously fit to the query and novel to the user. In the following section we discuss some of the shortcomings of this recommender sys-

tems approach, and describe an alternative interaction model for PQE that draws more closely on computational creativity techniques.

Beyond Recommendation: Narrative framing for behavioural suggestions

The novelty model we presented above is one component of the PQE architecture in Figure 1. We are in the process of developing a complete implementation containing personalisation of both novelty and value as well as a suggestion generator. In this section we discuss an approach to that suggestion generation component that goes beyond the “recommender systems” paradigm of providing a list of results. One problem with the recommendation approach is that it ignores the iterative nature of the PQE task. It is not possible to build *towards* any artefact or concept that might not be appreciable by the user given their current knowledge.

Result lists also do not typically provide a compelling framing for *why* each suggestion was made. Explanations have been incorporated into recommender systems (Tintarev and Masthoff 2011), but to our knowledge they have not explicitly been designed to be compelling or persuasive. We intend to compare the recommender systems model (providing an unordered, unframed set of suggestions) with a list of suggestions designed to be consumed sequentially, with each entry having accompanying framing. This framing will explain what the user should look for in each artefact, and what that will contribute to the overall goal of the sequence.

Our proposed system draws an analogy between this task of ordering and explaining suggestions and plot generation. Our system will be based on the Engagement-Reflection (ER) model, which has previously been employed to characterise plot generation (Pérez y Pérez and Sharples 2001), interior design (Pérez y Pérez, Aguilar, and Negrete 2010) and visual compositions (Pérez y Pérez, De Cossío, and Guerrero 2013), among other domains. In general terms, this model is composed of elements, relations between those elements, and actions that progress those relations. In the case of storytelling the elements are the characters in the narrative, between whom there exist emotional links and conflicts. Story-actions progress the tale by evolving those emotional links and tensions between characters. The advantage of this narrative analogy over viewing this as a planning problem is the ability to model *tension and interestingness*. A planning approach may produce a sequence of resources to satisfy the goal, but we hypothesise that a narrative generation approach could additionally *keep the user interested*.

In our application of the ER model to framing for behavioural suggestion elements will be the “concepts” within the domain, and their relationships will be determined by how those concepts interact within the user's past knowledge. The story actions are a sequence of artefacts, each of which introduces, relates, or elaborates on the user's knowledge about the concepts in the “plot”. The start of our “story” is the the user's current level of understanding of the domain. The culmination is the user appreciating a desired goal artefact that would not be comprehensible given their current level of knowledge. Framing will be generated

Title	URL	Unexpected topic combination
1: “Facility location with Service Installation Costs”	http://dl.acm.org/citation.cfm?id=982953	[<i>algorithm, problem, graph, comput+, time</i>] (43%) & [<i>application, service, mobile, provide+, resourc+</i>] (35%)
2: “Volunteer computing: the ultimate cloud”	http://dl.acm.org/citation.cfm?id=1734164	[<i>algorithm, problem, graph, comput+, time</i>] (39%) & [<i>comput+, will, student, learn, course</i>] (36%)
3: “Star Search: Effective Subgroups in Collaborative Social Networks”	http://dl.acm.org/citation.cfm?id=2810062	[<i>research, social, stud+, group, communit+</i>] (31%) & [<i>algorithm, problem, graph, comput+, time</i>] (26%)

Table 1: The three most unexpected paper abstracts in our database.

Title	URL	Expected topic combination
1: “A simple rewrite notion for call-time choice semantics”	http://dl.acm.org/citation.cfm?id=1273947	[<i>framework, structur+, specif+, approach, relat+</i>] (43%) & [<i>program, language, code, use, implement+</i>] (43%)
2: “Constrained kinds”	http://dl.acm.org/citation.cfm?id=2384675	[<i>program, language, code, use, implement+</i>] (43%) & [<i>framework, structur+, specif+, approach, relat+</i>] (42%)
3: “Slicing as a program transformation”	http://dl.acm.org/citation.cfm?id=1216375	[<i>program, language, code, use, implement+</i>] (50%) & [<i>framework, structur+, specif+, approach, relat+</i>] (42%)

Table 2: The three least unexpected paper abstracts in our database.

to explain each story action’s contribution towards the goal.

Engagement-Reflection for PQE suggestions

Figure 2 shows a schema of the ER model, with the engagement state on the top and the reflection state underneath. The process starts with a special instance of the Add Action process in which the initial world state – characters and their contexts – is added to the new action sequence. The model then iteratively probes its memory to recall parts of previous sequences that match, selects one action from a matching sequence to add, and then adds it. After a certain number of engagement cycles (3 in Mexica) reflection occurs and the emerging sequence is evaluated for interestingness, cohesion and whether it fits provided constraints. When no sufficiently similar stories exist in memory it enters Reflection to break the impasse, modifies the sequence, and then returns to Engagement.

In PQE we propose that this cycle could take place within the *suggestion generator* process, with the initial world state being provided by the novelty and value models of the user. ER requires a set of existing narratives that can be recalled during the process, which would need to be learnt or provided. A separate process would need to select a (currently unreachable due to being too novel) goal artefact for the ER model to strive towards. This process of inferring a goal from a particularly novel and/or valuable state parallels with

the model of specific curiosity in (Grace et al. 2017).

Inducing concepts for plot generation

Any implementation of the narrative suggestion process outlined above must grapple with defining what a “concept” is, how they will be learned, and how they will relate. In order to be useful for both PQE and ER, concepts are required to have certain properties:

1. Concepts must capture the themes that characterise artefacts within the domain (i.e. be useful for describing artefact meanings).
2. It must be possible to model the likelihood of concepts co-occurring so that they can be used to form the h-novelty model described in the PQE framework above.
3. Concepts must be comprehensible, so that the framing process can communicate them to the user.
4. Concepts must be algorithmically learnable from the data. Manual labelling or pruning may be appropriate, but the process should be mostly automated.
5. Concepts must have a notion of *accessibility* – what other concepts are sufficiently related that they should be appreciable by a user who comprehends this concept. This is required for the ER model to construct sequences of artefacts that lead to an eventual goal.

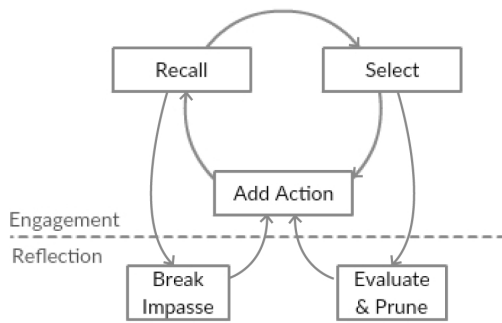


Figure 2: The processes of the ER model as it will be integrated with PQE. This cycle takes place inside the Suggestion Generator process.

From these properties we can see that the topic model representation described in this paper is likely insufficient. It satisfies the first, second and fourth properties, and perhaps the third as some methods for visually conveying topic models have been developed (Chuang, Manning, and Heer 2012). Topic models offer no simple way to describe what topics are adjacent or accessible to each other in terms of learnability. We are pursuing ways to develop a model of the concepts within research papers that also satisfies this fifth property. One possibility is to use the citation networks that exist between papers to characterise how new work builds on old. Another possibility is to identify what is mentioned when a paper is cited, and use that to infer what concepts a paper explains.

What determines “tension” for a framing narrative?

One role of the Reflection process in the ER model is to evaluate the interestingness of the emerging narrative by comparing it to a desired narrative structure. Narratives typically follow a degradation-improvement structure in which the situation gets worse and worse for the characters before a climactic moment after which things start to get better. This may happen cyclically, often with degradations of increasing magnitude. In ER this is called the tensional representation of a narrative.

In the PQE narrative framing task tension is not derived directly from the characters (the concepts in the artefacts being suggested), but from the user’s knowledge of those concepts. New concepts being introduced to the narrative raise the tension, as they can be reasonably be expected to increase the user’s confusion. Equally, concepts that are not currently connected to the other concepts within the story increase the tension, as the goal of these narratives is to render a single specific goal artefact comprehensible. Story actions (i.e. suggestions) that connect previously disparate concepts will decrease tension. We will also investigate modelling the tension between concepts that are (or seem) contradictory, although this will require a more detailed representation than the current topic models approach provides.

Conclusion

The PQE framework describes how computational creativity techniques might be used to encourage users to diversify their behaviour. The core principle of this framework is suggesting p-creative actions, based on the hypothesis that they will stimulate the user’s curiosity (their intrinsic motivation to explore the domain). The PQE framework is also an opportunity to apply computational creativity techniques outside of domains that are traditionally thought of as “creative”. The approach applies to domains, such as education and nutrition, where divergent behaviour is beneficial.

We have prototyped an h-novelty model, one component of the PQE system, in the domain of research paper recommendation. We model novelty in unstructured text documents using topic models, a machine learning approach that induces the dominant “themes” of a database and describes how they relate. We base our novelty model on papers that exhibit unexpected combinations of these topics. Our prototype, which operates only on research paper abstracts rather than the full text, can successfully identify abstracts that are highly novel as well as those that are highly conventional. We are working on developing this novelty model into a recommender-based system that can suggest papers that are simultaneously fit to the user’s search queries, and surprising according to their knowledge of the domain.

We also describe our plans to incorporate this novelty model into a system that provides a specific sequence of artefact suggestions along with explanations of why the user should engage with each. This is based on treating the p-creativity stimulating suggestion task as one of narrative generation. We propose a approach based on the ER model that describes how such a system could construct a compelling journey through the domain, building at each step the user’s capacity to appreciate a goal artefact.

In summary, our key contribution in this paper is an approach to suggesting unexpected research papers by identifying novel combinations of topics. We show how we can select documents that include topic combinations that have a very low probability of occurring together. We use a bag of words representation for each paper and the Correlated Topic Model algorithm to generate the distribution and correlation of topics across the corpus of research papers. We propose that this core element of our co-creative system has the potential to deliver surprising research papers to a student at a minimum, and beyond that to provide a plan for generating a surprising research paper by suggesting unusual combinations of topics for new research papers.

References

- Amabile, T. 1996. *Creativity in context*. Westview press.
- Baldi, P., and Itti, L. 2010. Of bits and wows: a bayesian theory of surprise with applications to attention. *Neural Networks* 23(5):649–666.
- Barto, A. G.; Singh, S.; and Chentanez, N. 2004. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, 112–19. Citeseer.

- Berger, W. 2010. *CAD Monkeys, Dinosaur Babies, and T-Shaped People: Inside the World of Design Thinking and How It Can Spark Creativity and Innovation*. Penguin.
- Berlyne, D. E. 1966. Curiosity and exploration. *Science* 153(3731):25–33.
- Blei, D. M., and Lafferty, J. D. 2007. A correlated topic model of science. *The Annals of Applied Statistics* 17–35.
- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.
- Boden, M. A. 2004. *The creative mind: Myths and mechanisms*. Psychology Press.
- Charnley, J.; Pease, A.; and Colton, S. 2012. On the notion of framing in computational creativity. In *Proceedings of the Third International Conference on Computational Creativity*, 77–81.
- Chuang, J.; Manning, C. D.; and Heer, J. 2012. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 74–77. ACM.
- Corneli, J.; Jordanous, A.; Guckelsberger, C.; Pease, A.; and Colton, S. 2014. Modelling serendipity in a computational context. *arXiv preprint arXiv:1411.0440*.
- Csikszentmihalyi, M., and Rathunde, K. 1992. The measurement of flow in everyday life: toward a theory of emergent motivation. In *Nebraska Symposium on Motivation*, volume 40, 57–97.
- Fogg, B. J. 2002. Persuasive technology: using computers to change what we think and do. *Ubiquity* 2002(December):5.
- Fogg, B. J. 2009. A behavior model for persuasive design. In *Proceedings of the 4th international Conference on Persuasive Technology*, 40. ACM.
- Grace, K., and Maher, M. L. 2014. What to expect when you're expecting: the role of unexpectedness in computationally evaluating creativity. In *Proceedings of the 4th International Conference on Computational Creativity*, to appear.
- Grace, K.; Maher, M. L.; Fisher, D.; and Brady, K. 2015. Modeling expectation for evaluating surprise in design creativity. In *Design Computing and Cognition'14*. Springer International Publishing, 189–206.
- Grace, K.; Maher, M. L.; Wilson, D.; and Najjar, N. 2017. Personalised specific curiosity for computational design systems. In *Design Computing and Cognition'16*. Springer, 593–610.
- Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22(1):5–53.
- Kay, J. 1994. The um toolkit for cooperative user modelling. *User Modeling and User-Adapted Interaction* 4(3):149–196.
- Kidd, D. C., and Castano, E. 2013. Reading literary fiction improves theory of mind. *Science* 342(6156):377–380.
- Klasnja, P.; Consolvo, S.; and Pratt, W. 2011. How to evaluate technologies for health behavior change in hci research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3063–3072. ACM.
- Merrick, K. E., and Maher, M. L. 2009. *Motivated reinforcement learning: curious characters for multiuser games*. Springer Science & Business Media.
- Moran, T. P., and Carroll, J. M. 1996. *Design rationale: concepts, techniques, and use*. L. Erlbaum Associates Inc.
- Pérez y Pérez, R.; Aguilar, A.; and Negrete, S. 2010. The eri-designer: A computer model for the arrangement of furniture. *Minds and Machines* 20(4):533–564.
- Pérez y Pérez, R., and Sharples, M. 2001. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence* 13(2):119–139.
- Pérez y Pérez, R.; De Cossío, M. G.; and Guerrero, I. 2013. A computer model for the generation of visual compositions. In *Proceedings of the Fourth International Conference on Computational Creativity*, 105. Citeseer.
- Purcell, A. T., and Gero, J. S. 1996. Design and other types of fixation. *Design studies* 17(4):363–383.
- Reio, T. G., and Wiswell, A. 2000. Field investigation of the relationship among adult curiosity, workplace learning, and job performance. *Human Resource Development Quarterly* 11(1):5–30.
- Ricci, F.; Rokach, L.; and Shapira, B. 2011. *Introduction to recommender systems handbook*. Springer.
- Roberts, M. E.; Stewart, B. M.; and Tingley, D. 2014. stm: R package for structural topic models. *R package version 0.6.1*.
- Runco, M. A., and Richards, R. 1997. *Eminent creativity, everyday creativity, and health*. Greenwood Publishing Group.
- Ryan, R. M., and Deci, E. L. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* 25(1):54–67.
- Saunders, R., and Gero, J. S. 2001. Artificial creativity: A synthetic approach to the study of creative behaviour. *Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney* 113–139.
- Schmidhuber, J. 2010. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development* 2(3):230–247.
- Suwa, M.; Gero, J.; and Purcell, T. 2000. Unexpected discoveries and s-invention of design requirements: important vehicles for a design process. *Design studies* 21(6):539–567.
- Tintarev, N., and Masthoff, J. 2011. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*. Springer, 479–510.
- Vadiveloo, M.; Dixon, L. B.; Mijanovich, T.; Elbel, B.; and Parekh, N. 2015. Dietary variety is inversely associated with body adiposity among us adults using a novel food diversity index. *The Journal of nutrition* 145(3):555–563.

Addressing the “Why?” in Computational Creativity: A Non-Anthropocentric, Minimal Model of Intentional Creative Agency

Christian Guckelsberger¹, Christoph Salge^{2,4} and Simon Colton^{1,3}

¹Computational Creativity Group, Goldsmiths, University of London, UK

²Adaptive Systems Research Group, University of Hertfordshire, UK

³The Metamakers Institute, Falmouth University, UK

⁴Game Innovation Lab, New York University, USA

c.guckelsberger@gold.ac.uk

Abstract

Generally, computational creativity (CC) systems cannot explain *why* they are being creative, without ultimately referring back to the values and goals of their designer. Answering the “*why?*” would allow for the attribution of *intentional agency*, and likely lead to a stronger perception of creativity. Enactive artificial intelligence, a framework inspired by autopoietic enactive cognitive science, equips us with the necessary conditions for a value function to reflect a system’s own intrinsic goals. We translate the framework’s general claims to CC and ground a system’s creative activity intrinsically in the maintenance of its identity. We relate to candidate computational principles to realise enactive artificial agents, thus laying the foundations for a minimal, non-anthropocentric model of intentional creative agency. We discuss first implications for the design and evaluation of CC, and address why human-level intentional creative agency is so hard to achieve. We ultimately propose a new research direction in CC, where intentional creative agency is addressed bottom up.

Introduction

Imagine conducting an interrogation experiment, in which human participants are to judge the creativity of a state of the art computational creativity (CC) system. The system could be a piece of software or consist of one or several embodied agents, it could act in the lab or in the field, and there is no restriction on the type of creativity exercised. Crucially, the system has unlimited capacities to enter into a dialogue and to frame (Charnley, Pease, and Colton, 2012) its actions. Participants include the general public, CC researchers, as well as expert practitioners and critics of the type of creativity exercised. In contrast to the Turing (1950) test, the system must always answer truthfully.

We would expect most participants to base their judgement on the system’s observed behaviour and produced artefacts only. Some might make few inquiries about the system’s process, while others might engage in a deep interrogation. We would certainly end up with divided opinions on the creativity of the system, confirming the view that creativity is an *essentially contested concept* (Gallie, 1955; Jordanous and Keller, 2016). While we would expect most participants to attribute creativity to the system if its behaviour and output was *novel* and *valuable*, others might be more inquisitive, and eventually fail the system because it cannot give satisfactory answers to *why it acted the way it did*.

This addresses the system’s *intentional agency*, i.e. its capacity to have a purpose, goal or directive for creative action (cf. Ventura, 2016). However, we doubt that any existing CC system, even with our hypothetical dialogue capacity, could answer questions about its intentionality without referring to its designer’s goals. Jordanous and Keller (2016) have empirically identified intentionality as one factor in the perception of creative systems. We believe that a system’s inability to account for its *own* intentionality is a valid reason for people to disapprove it of being creative, particularly creative *in its own right*. We also doubt that these systems fully own their artefacts, as they cannot justify why they *originated* them. Ada Lovelace famously addressed originality:

“The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform”. (Menabrea and Lovelace, 1842)

By stating that the Analytical Engine has no *pretensions* to originate anything, Lovelace gives us the key to what we believe is the answer to the “*why?*” in CC: if we want to design systems that are deemed creative in their own right, we need them to *own their goals*. Their pretensions, i.e. their motivations, must not be the designers’, but arise from their own, genuine *concern*. This concern forms the basis of a system’s *sense-making*, i.e. the assignment of values to features of the world that are of relevance to the system itself. To be considered an intentional *agent* in its own right, it must use these values as the basis of action.

Not only do existing implementations fail to address this ultimate challenge of intentional creative agency – CC also misses a theoretical framework describing the conditions for intrinsic goal-ownership underlying intentionality. We believe that the development of such a framework is hindered by CC’s focus on human creativity in system design and evaluation. Human creativity unfolds within a complex network of influences shaped by a person’s social and cultural environment (Bown, 2015; Jordanous, 2015). Identifying why a person was being creative and translating the findings to formal models therefore is hard. We also believe that CC’s focus on big-C artefacts (Kaufman and Beghetto, 2009) is detrimental, as the values within are hard to disentangle and invite complex interpretations of the notion of creativity. Despite these impediments, CC’s major contributions to key concepts around intentionality such as *adaptivity*

(Bown, 2012) and *agency* (Bown, 2015) are still strictly anthropocentric. The concept of creativity is human-made, but it should not remain human-centric: by understanding how intentional creative agency can be brought about in artificial agents, we can identify creativity in systems that previously remained unnoticed, learn about new forms of creative behaviour, and actively support their emergence.

We adopt Froese and Ziemke's (2009) *enactive artificial intelligence* (AI) framework, which provides a non-anthropocentric account of intentional agency. In contrast to Dennett's (1989) *intentional stance* which could be applied to any system, Froese and Ziemke (2009) limit intentionality to systems that share some essential characteristics with living organisms. Based on the bio-systemic foundations of autopoietic enactive cognitive science, they argue that the *purpose* of an intentional agent, determining its intrinsic goals, is the maintenance of its existence. They propose two conditions for intentional agency and sense-making in artificial agents: *constitutive autonomy* and *adaptivity*.

We argue that adaptive and constitutively autonomous agents must necessarily exhibit behaviour which many would deem creative. More specifically, we claim that two forms of creativity, *autopoietic*- and *adaptive creativity*, are intrinsic to enactive artificial agents. We hypothesise how our minimal model could give rise to more complex forms of creative behaviour, and briefly outline computational principles to put our theoretical considerations into practice. We thus extend Froese and Ziemke's (2009) framework with an account for creativity to establish a non-anthropocentric, minimal model of intentional creative agency. Our findings suggest that creativity can be found in any living being, not only in humans and highly developed animals. We discuss first implications of our enactive account for the perception of creativity in nature, for CC evaluation, and for building artificial agents with human-level creativity.

Our model is non-anthropocentric but agent-centric, looking at the value of actions and artefacts from the perspective of the creative agent, in contrast to an external observer. It is minimal in that we account for intentional agency in p-creative behaviour (Boden, 2003) at the *edge of being*, in contrast to big-C, h-creativity shaped in a social context (Saunders and Bown, 2015). Accepting creativity as essentially contested, we encourage notions of creativity without or with externally attributed values; by identifying what is required for intentional creative agency, we do not constrain, but widen the scope of what should be considered creative.

Our ultimate goal is to propose a *new direction for research* in CC, in which intentional creative agency is addressed from the bottom up. We motivate our approach by asking the "*why?*" for existing CC systems with a focus on big-C creativity. However, we do not address how this question could be answered in terms of communication and framing (Charnley, Pease, and Colton, 2012). While we put forward a hypothesis about climbing up the *creativity ladder*, closing the explanatory gap between our model and human-level, big-C creativity is subject to further research and experimentation. We agree with Froese and Ziemke (2009) that their conditions are necessary for intentional agency, but also share their caution that they might not be sufficient.

Climbing Down the Creativity Ladder

Traditionally, much research in CC is software-based and models complex human-level creativity in artistic domains. We show by means of case-studies and by reference to traditional AI arguments that this symbolic approach cannot possibly account for intrinsic goal-ownership. Embracing the paradigm of embodied and situated cognition reduces this challenge, but is still insufficient. Our solution eventually leads us away from human, big-C artefacts down to creative behaviour in minimal agents with a precarious existence.

Symbolic Computational Creativity

Developed for more than a decade, *The Painting Fool* (TPF) is a prime example of a symbolic CC system. The project's goal is to create a system which is eventually taken seriously as a creative artist in its own right (Colton, 2012). TPF was used extensively in field studies about the perception of creativity by unbiased observers. In the "You Can't Know my Mind" exhibition (Colton and Ventura, 2014), the software accompanied its portraits with a commentary on their creation, enabling visitors to project intentionality on the system. We use this context for a case-study in which TPF is equipped with our hypothetical dialogue system.

Being asked "*Why* did you paint my portrait in that way?", TPF could genuinely answer: "Because I was reading the following newspaper article, and this was used to simulate a mood which drove the artistic choices I made". Digging deeper, the next question could be: "*Why* were you reading the newspaper?". The answer to this would be: "Because my programmer told me to do so". A particularly curious participant might then ask: "*Why* did your programmer tell you to read the newspaper?", to which the system would respond: "So that I have an interesting backstory for my creative acts". Asked "*Why* do you need such a backstory?", the system's honest answer would be "So that I appear more creative". We see that our persistent questioning yields a circularity with the reason behind certain behaviours being to promote the appearance of creativity. It is fair to say that TPF was given the ability to sustain the impression of intentionality at only *the first level*.

Cook and Colton (2015) account for successful answers one level below. To overcome randomness and hard-coding, they introduce a method to invent distinct, but consistent and therefore believable preferences. Being asked why it painted a portrait in a certain tone, TPF could truthfully explain its behaviour through a set of initially generated colour preferences residing within the system. However, asking why it came up with a certain set of preferences, or with the constraints for their consistency, we would again end up in circularity. We believe that many would consider this circularity an unsatisfactory answer to the "*why?*" in CC, and potentially not attribute creativity to such a system.

By asking the "*why?*" for the prototype algorithms described by Ventura (2016), we find that this shortcoming applies to symbolic CC in general. In CC, being intentional is understood as having a goal or directive for action. These goals are modelled with value functions, used in action selection and the post-hoc evaluation of artefacts. There is

agreement that in human creativity, such values do not come from the creative person alone, but from a network of human influences (cf. Bown, 2015). In symbolic CC however, a system's goals come exclusively from *other human actors*, and do not reside *in the system itself*: TPF's goals are determined by the algorithms and constraints specified by its designer, and its simulated mood depends on the author of the newspaper article it analyses. Value functions in symbolic CC do not reflect a system's own goals. Moreover, they typically reflect the designer's goals in *respect to a particular artefact*. The system's purpose is determined by the purpose of the artefacts produced (cf. Gervás and León, 2016). The concept of intentionality in symbolic CC is thus very weak.

One might argue that CC software simply does not go "deep enough", but its symbolic nature makes it fundamentally incapable of intrinsic goal-ownership. These systems are *computationalist*, in that creativity is reduced to the manipulation of symbols. Computationalism is subject to a range of classic AI problems such as the *frame problem* (Wheeler, 2005, p. 179) and the *symbol grounding problem* (Harnad, 1990). Searle (1980) addresses the latter in his famous *Chinese Room argument*, showing that syntax is not sufficient for semantics. By translating Searle's argument to artefact creation, Al-Rifaie and Bishop (2015) show that it also applies to CC. Because symbolic CC cannot ground meaning, it also cannot give rise to goals which are meaningful from the system's own perspective.

Embodied Computational Creativity

Brooks (1991) has challenged these problems of symbolic AI by embracing the ideas of situated and embodied cognition. In situated cognition, cognitive processes emerge from the interaction of an organism and its world, and are thus inseparable from action. Embodied cognition emphasises the role of an agent's physical body in shaping cognitive processes. Potentially influenced by *systems theories of creativity* (cf. Saunders, 2012), CC has adopted the embodied and situated approach: there is general agreement that creativity does not occur in a vacuum: it is a *situated* activity, in that it relates to a cultural, social and personal context. However, it is also physically conditioned on an agent's *embodiment* and structured by how an agent's morphology, sensors and actuators shape its interaction with the world.

Embodied AI has developed into a mature framework for modelling artificial agents, and Pfeifer, Iida, and Bongard (2005) describe its characteristics via a list of design principles. Most importantly, they require an agent to have a value function, telling it whether an action was good or bad. The agent must then use these values to motivate its *behaviour*. Embodied AI's value principle thus operates one level below the value function in symbolic CC, which is primarily used in *artefact evaluation* to assess the success of generative routines. We can see the embodied AI principles being adopted in CC: Hoffman and Weinberg (2010) for instance leverage the effect of an agent's morphology on creativity. Their robot Marimba player *Shimon* improvises in real-time to a human pianist's performance. In contrast to symbolic CC, music here is not understood as a sequence of notes, but as a choreography of movements constrained by the robot's

morphology. Embodied AI counteracts the Lovelace objection in that it demands a reduction of the designer's influence to foster emergent behaviour. Saunders et al. (2010) investigate the emergence of autonomous and creative behaviour from the interaction of agents in *Curious Whispers*, where a society of simple robots generate and listen to tunes.

Embodied AI practitioners such as Dreyfus (1992) claim that the symbol grounding problem can be overcome by embedding agents in a closed sensorimotor loop: an agent perceives the effects of its actuators on the external environment which, via its internal controller, lead to the next action. An embodied agent can avoid the use of internal symbolic representations, by using "the world as its own model" (Brooks, 1991). However, Froese and Ziemke (2009) argue that this only solves the first part of the *frame problem*:

"Given a dynamically changing world, how is a non-magical system (...) to take account of those state changes (...) and those unchanged states in that world that matter, while ignoring those that do not? And how is that system to retrieve and (if necessary) to revise, out of all the beliefs that it possesses, just those beliefs that are *relevant* in some particular context of action?" (Wheeler, 2005, p. 179, emphasis added)

They argue that being embedded in a closed sensorimotor loop is not sufficient for an agent to evaluate features of the world relative to its *own* purpose. Embodied AI's value principle is at the centre of their criticism, as it does not preclude the external assignment of values. *Shimon*'s goals for instance are hard-coded, allowing the system to perform a prescribed set of interactions to support the human musician. Once its counterpart deviates from the protocol, the system fails to operate. *Shimon* does not act for its own purpose.

Intrinsic Motivation to the Rescue?

We can say that *Shimon* is *extrinsically motivated*, as its goals, defining its behaviour, are imposed by its designers. In contrast, agents can also be intrinsically motivated, performing "an activity for its inherent satisfaction rather than for some separable consequence" (Ryan and Deci, 2000). This psychological definition has been complemented with computational approaches, surveyed by Oudeyer and Kaplan (2008). Per definition, these approaches have to rely on agent-internal experience alone, based on the semantic-free relationship between sensors and actuators. Existing models capture drives like *learning progress* or *curiosity*. *Curious Whispers* uses a model of curiosity to make robots seek interesting tunes. Here, interestingness is quantified by mapping a new tune's novelty, relative to past experience, on a Wundt curve (Saunders et al., 2010). The robots thus listen to tunes which are neither too similar, nor too different to those previously experienced. If the tunes of other agents are not interesting enough, the robots create their own.

Formal models of intrinsic motivation ground behaviour in an agent's sensorimotor loop, and thus partly overcome the criticism of embodied AI's value function. Many models of intrinsic motivation are bio-inspired, but can we claim that any intrinsic value function and the emerging behaviour relates to an agent's own purpose? Why would a certain

agent be curious and seek for new stimuli, instead of hiding in a dark room? When does a particular motivational model reflect the system’s intrinsic goals, not the designer’s? We could simply assume agents to share our goals and behave like us. However, if we accept that cognition and thus behaviour is shaped by our embodiment and situatedness, there is no justification for this anthropocentric stance. The previously outlined theories are not sufficient to decide whether an action policy, which formally qualifies as an intrinsic motivation, reflects the intrinsic goals of a given agent.

Enactive Computational Creativity

Embodied and situated cognition overcomes some shortcomings of symbolic CC, but does not account for an agent’s own purpose, i.e. *intrinsic teleology*. Despite this, we argue that intentional creative agency is not an infinite regress problem – we can resolve the circularity demonstrated in symbolic and embodied CC by grounding an agent’s actions in a model of intrinsic motivation, based on a *suitable* intrinsic value function. In their enactive AI framework, Froese and Ziemke (2009) introduce the missing conditions for such a value function to relate to an agent’s own goals. However, their framework focusses on cognition in general. We argue that enactive agents necessarily have to exhibit two specific forms of creativity which can potentially give rise to complex creative behaviour, and thus establish an account of intentional creative agency. We refer to candidate principles to realise these theoretical conditions, hence laying the foundations for a model of *enactive computational creativity*¹.

Enactive Artificial Intelligence

Froese and Ziemke identify the necessary requirements for an intrinsic motivation to reflect an agent’s own goals, by looking at how living beings are different from the non-living with respect to their *purpose*. As mainstream biology offers no distinction in respect to purpose, they draw on the biosystemic foundations of *enactive cognitive science*.

Enactivism is a non-reductive, non-representationalist theory of cognition which adopts the embodied and situated paradigm, but additionally grounds cognition in practical activity. Following O’Regan and Noë’s (2001) theory of sensorimotor contingencies, Noë (2004) stresses that what we perceive is determined by what we do. At the core of enactivism thus is the idea that individuals do not passively create internal representations of a pre-given external world (Stewart, 2010); through their interaction with the environment, agents enact, i.e. actively construct, their own world of significance (Varela, Thompson, and Rosch, 1991). Enactivism thus roots sense-making in action. Froese and Ziemke follow the autopoietic branch of enactivism which formulates a strong life-mind continuity, and explicitly addresses intentional agency and sense-making (cf. Thompson, 2004).

¹With “computational”, we relate to CC as a research field, not to computationalism. Our model uses insights from autopoietic enactivism to guide the design and evaluation of artificial, intentionally creative agents. We believe that this approach is not restricted to simulated embodied systems and robotics, but could also inform other means of creating artificial agents, e.g. synthetic biology.

At the core of autopoietic enactivism are three theories which root intentional agency in the living, and thus shed light on our missing conditions. Kant argues that living systems have a “natural purpose” in that they are “both cause and effect in themselves” (Kant, 1995, §64). He suggests that the intrinsic goal-directedness of living beings arises from their purpose to self-produce (cf. Weber and Varela, 2002). The biologist von Uexküll translates these findings to sense-making by arguing that living beings, through their sensorimotor activity, construct their own, unique perspective on the world based on the requirements of their self-production (Di Paolo, 2003). In what he refers to as *Umwelt*, features of the environment are captured and assigned significance in terms of how they affect the individuals’ self-organisation and ongoing preservation. The bio-philosopher Jonas finally provides us with an account of *identity* (Jonas, 1982, p.126). He argues that simple matter and most artificial systems exist without the need or capacity to act. Living beings in contrast have a *precarious existence*, in that they could at any time become a non-being; they have to continuously interact with their environment in order to satisfy their material and energetic requirements. Based on these theoretical underpinnings, Froese and Ziemke (2009) claim that an agent’s value function must reflect the concern about its maintenance of identity. Similarly Haselager (2007) argues that “only systems with a self-generated identity can be said to genuinely own and enact their own goals”.

The notion of self-production is operationalised by Maturana and Varela’s (1987) concept of *autopoiesis*. An autopoietic system represents a minimal living organisation which realises constitutive autonomy, as it physically individuates an entity from its environment, and constitutes its identity in the domain. The concept of autopoiesis only applies to biochemical systems, and is generalised by the notion of *organisational closure*. A system implementing organisational closure is a network of processes that generate and sustain its identity under precarious conditions, and that form a unity in a containing domain (Varela, 1979). The first condition for enactive artificial agents states that intrinsic teleology requires constitutive autonomy:

EAI-1 (Constitutive autonomy): “the system must be capable of generating its own systemic identity at some level of description” (Froese and Ziemke, 2009).

This condition represents the enactive version of embodied AI’s value principle, but it is strictly intrinsic in that the agent must relate value to the maintenance of its own precarious existence. However, Froese and Ziemke (2009) argue that this alone is not sufficient to maintain an agent’s identity over the long term: in a dynamic and uncertain environment, an agent must be able to compensate for unexpected events. This requires a value function to distinguish external events more gradually relative to the agent’s organisation. Nevertheless, the concept of organisational closure per se is only binary: a system either maintains its organisational closure or not. Di Paolo (2005) compensates for this limitation by presupposing the existence of a *viability* set, i.e., levels of structural change that allow living beings to “sustain a certain range of perturbations [...] before they lose their au-

topoiesis” (Di Paolo, 2005). He defines *adaptivity* as

“a system’s capacity, in some circumstances, to regulate its states and its relation to the environment with the result that, if the states are sufficiently close to the boundary of viability,

1. Tendencies are distinguished and acted upon depending on whether the states will approach or recede from the boundary and, as a consequence,
2. Tendencies of the first kind are moved closer to or transformed into tendencies of the second and so future states are prevented from reaching the boundary with an outward velocity”. (Di Paolo, 2005)

The necessity of adaptivity for intentional agency is covered by the second condition for enactive agents:

EAI-2 (Adaptivity): “the system must have the capacity to actively regulate its ongoing sensorimotor interaction in relation to a viability constraint” (Froese and Ziemke, 2009).

In summary, enactive AI complements and extends embodied AI’s approach to move sense-making into the sensorimotor loop, by grounding sensorimotor interaction in an agent’s maintenance of its identity. Froese and Ziemke (2009) claim that an intentional agent must not only be embodied, but also realise constitutive autonomy and adaptivity via its value function and action policy.

Creativity at the Edge of Being and Beyond

Two factors can be frequently observed in the attribution of creativity to other humans: novelty and value (Paul and Kaufmann, 2014; Jordanous and Keller, 2016). Current CC systems lack intrinsic goal-ownership, because the discipline misses an account for value from a non-human perspective. When researchers talk about creativity outside the human domain, they simply drop value from the definition: What Boden (2015) labels “biological creativity” in the context of artificial life is earlier defined by Bown (2012) as “generative creativity”: a system’s ability to create “new patterns or behaviours regardless of the benefit to that system”.

Autopoietic enactivism provides us with an intrinsic account of value in any constitutively autonomous system. Although autopoiesis literally means “self-creation”, we argue that a system with organisational closure alone cannot be assumed to exhibit *novel* behaviour in a non-trivial sense. We thus distinguish *autopoietic creativity* in organisationally closed systems from *adaptive creativity* in fully enactive agents. We claim that an enactive AI must necessarily exhibit adaptive creativity. The notion “adaptive creativity” has been used loosely in creativity studies (Kirton, 1994) and CC (Bown, 2012) before; we make it more concrete by drawing on Di Paolo’s (2005) definition of adaptivity. While the notion of value in human-creativity is ambiguous and relates to complex concepts such as interestingness and aesthetics, enactive AI allows us to root it in utility alone.

Autopoietic Creativity Autopoiesis is generalised in operational closure. Varela (1984) uses the notion of a “creative circle” to describe both (i) the self-organisation of an organisationally closed system, and (ii) its self-maintenance

as an autonomous unity. We can distinguish two notions of creativity along these stages. While we witness an autonomous system emerging out of something else, e.g. a cell out of a molecular soup, we can only apply the notion of *generative creativity*. From the perspective of an external observer, the system appears as a transient, ever-changing artefact. At this stage, there is no perspective of the system itself yet, and value can only be externally imposed.

However, once the system has individuated itself from the containing domain, it establishes a unique perspective on the world, mediated by its situatedness and embodiment. We can now consider creativity from the system’s *own* perspective. From here, every change to its structure has a value, in that it either preserves or destroys its organisation. To maintain its identity, a system has to engage in positively valued, organisation-preserving operations (Jonas, 1982, p. 72). We define *autopoietic creativity* as a system’s *active* modification of structure to ensure its continuous existence. A system cannot be referred to as autopoietically creative, if these changes are caused exclusively by external forces.

We argue that this form of creativity is very minimal, in that the exhibited behaviour is not necessarily novel in a non-trivial sense. To distinguish autopoiesis clearly from adaptivity, we constrain our claim to a fictional autopoietic system operating free from perturbations, i.e. there are no external forces which could destroy its organisation. For a system to maintain organisational closure, it is sufficient to engage in a cyclic flow of material configurations. In the absence of perturbations, novelty in the system’s change of structure only depends on its current shape and its internal dynamics. Consequently, the range of possible changes and thus novelty could be fully pre-specified and would be quickly exhausted. This does not mean that an autopoietic system could not exhibit valuable and novel behaviour; but since this is not required by definition, we cannot assume it to be creative in the popular sense described earlier.

Adaptive Creativity The notion of *autopoietic creativity* is rather theoretical, as a physically embodied system will always be subject to entropic forces, either implicitly via material and energetic dependencies on the environment, or explicitly through perturbations leading to its disorganisation. To maintain its identity, such a system has to be adaptive. We argue that adaptivity represents the essential mechanism for creative behaviour in an autopoietic system.

According to Di Paolo (2005), an adaptive system does not disintegrate in a second, but can undergo a variety of structural changes before it loses its existence. These structural changes characterise the system’s *viability set*. Di Paolo argues that an adaptive system must be able to recognise whenever it is moving closer to its viability boundary, and either slow this tendency down, or invert it in order to be more robust against future perturbations. In contrast to the earlier isolated system, the future structure of an adaptive system is not only affected by its current shape and internal dynamics, but also by external perturbations. An adaptive system exhibits *novel* behaviour when it is either (i) responding to a familiar perturbation in a different way than before, or when it is (ii) responding to a previously unen-

countered perturbation. This is non-trivial as in a dynamic environment, an embodied system cannot be hard-wired to anticipate and defend its identity successfully against any possible perturbation; increasing the complexity of its internal dynamics comes with an increase in energetic and material requirements, and thus counteracts viability. Any constitutively autonomous and adaptive agent must be able to respond flexibly to potentially unencountered perturbations in novel but valuable ways. We define this as *adaptive creativity* and conclude that an enactive agent with intentional agency must necessarily be adaptively creative.

Moving Away From the Edge An adaptive system must be able to evaluate each structural change in its viability set relative to its viability boundary. A structural change that would move the system closer to its viability boundary would be valued negative and vice-versa. Adaptive creativity, i.e. responding with novel and organisation-preserving actions to potentially unencountered perturbations, allows a system to move away from the boundary.

However, Di Paolo (2005) only requires an adaptive system to regulate its states “in some circumstances”. While such a system would likely stay close to its viability boundary, acting *consistently* creative would allow it to move away from the edge of being. We hypothesise that such *aspirational creative behaviour* requires, but also gives rise to more complex forms of creativity. To compensate and escape viability tendencies, a system could change its behaviour via sensorimotor coordination, but it could also adapt and augment its morphology, or change its environment. As Gibson has stated: “Why has man changed the shapes and substances of his environment? To change what it affords him. He has made more available what benefits him and less pressing what injures him” (Gibson, 1986, p. 123). Furthermore, we expect sociality to be particularly important for sustaining viability. We hypothesise that aspirational adaptive creativity allows us to climb up the *creativity ladder* again. However, detailed theoretical work and experimentation is subject to future work.

Operational Principles

There are many models of artificial curiosity, and we cannot make a general claim without evaluating them individually based on the enactive AI conditions. We believe that the intrinsic model of curiosity in *Curious Whispers* (Saunders et al., 2010) cannot give rise to intentional creative agency. Only in the presence of a function indicating the agent’s viability relative to its current state can an agent relate its behaviour to the maintenance of its existence. An agent is not constitutively autonomous, if it does not act based on how close it is to losing its autonomy. In *Curious Whispers*, it is unclear how selecting tunes which are neither too familiar nor too novel relates to a robot’s viability.

Recently, several candidate principles were hypothesised to realise the enactive AI conditions. In his *Free Energy Principle*, Friston (2010) argues that in order to not become disorganised, living organisms and artificial agents have to maintain an upper bound on the entropy of their sensory states as the average of surprise. The agent’s free energy, for-

malised as the difference between its model and the world, constitutes a tractable upper bound on surprise. Allen and Friston (2016) argue in the context of predictive coding that a free energy minimising agent can be considered as enactive, and that it realises constitutive autonomy and adaptivity. The *Free Energy Principle* evolves around action and model optimality, but it is unclear whether it provides values for sense-making. Friston’s principle appears conceptually close to maximising *predictive information* (Ay et al., 2008), i.e. the information an agent’s past states hold about its future. A comparison of the principles is yet to be done.

Guckelsberger and Salge (2016) argue that the information-theoretic principle of *empowerment maximisation* fulfils the enactive AI conditions. Empowerment quantifies the efficiency of an agent’s sensorimotor loop. Given that an agent could not counteract perturbations and satisfy its energetic or material requirements with a dysfunctional loop, they argue that empowerment represents a proxy to the agent’s viability, and that maintaining it realises organisational closure. They show via simulations that an agent which maximises empowerment in its action policy realises adaptivity. Crucially, an empowerment maximising agent not only adapts sporadically, but consistently increases its viability. We thus consider this a promising candidate to investigate whether consistent adaptive creativity leads to more complex creative behaviour.

Implications: The Embodiment Distance

We have proposed a model of intentional creative agency in which value is grounded in a system’s maintenance of its precarious existence. We argue that a system’s sense-making and thus creative behaviour is determined by its embodiment, which is usually very *different from ours*: a physically embodied agent can have a different morphology, a different access to the world through its sensors and actuators, and other energetic and material dependencies. We briefly discuss first implications of this *embodiment distance* for the perception of creativity in nature and artificial systems, as well as for the design of human-like CC.

CC field studies (e.g. Colton and Ventura, 2014) demonstrate that there are many systems that unbiased observers deem creative, although these systems ultimately do not act creatively in respect to their own goals. This is fair, as creativity is an essentially contested concept. Here, we look at the opposite case: we argue that there are many *adaptively creative* systems which perform novel and valuable actions relative to their intrinsic values, but would *not* be deemed creative. These systems root their values and thus behaviour in their embodiment. Their artefacts, i.e. their own structure and marks in the environment, are consequently value-laden relative to their embodiment. Our judgement of human artefacts is sensitive to our human embodiment, as psychological experiments in embodied aesthetics (Johnson, 2008) suggest. When we evaluate the creativity of non-human systems with intentional agency, we are likely to misjudge value in their behaviour or artefacts, or hesitate to attribute any value at all, as our embodiment distance is too large. This means that we are likely to misjudge or even fail to acknowledge the adaptive creativity of some systems, while agents of the

same type would value it highly. To judge the adaptive creativity of a system with creative intentional agency, we need to take the perspective of that system, and assess its sense-making and behaviour from there.

The embodiment distance is also relevant for the design of intentional CC agents with human-like creativity. Dreyfus (2007) has argued that human-like cognition in artificial agents requires us to replicate human embodiment. While this is not an issue if we are only interested in realising adaptive creativity in minimal intentional agents, Dreyfus' claim remains critical for reproducing human-like creativity.

Related Work

We have operationalised the previously loose notions of agency (Bown, 2015), adaptivity (Bown, 2012) and autonomy (Saunders, 2012) in CC by drawing on autopoietic enactivism. We seem to be the first to embrace this branch of enactive cognitive science in CC; Davis et al. (2015) develop a model of creative collaboration and co-creation based on Noë's (2004) sensorimotor enactivism; however, the sensorimotor branch focusses on the constitutive role of action in perception, but misses an account of intentional agency.

The concept of autopoiesis has been used in systems theories of creativity. However, it has been employed rather metaphorically (Gornev, 1997) or to describe creativity in society, not in individual agents (Iba, 2010). CC has adopted the concept of autopoiesis: Bishop and Al-Rifaie (2016) implemented an autopoietic model of creativity as a swarm intelligence system, but they specify their value function explicitly, instead of using an intrinsic account of sense-making. Saunders (2012) investigates the role of communication for autonomy in creative agent societies, but does not ground the behaviour of individual agents in the maintenance of their identity. Most importantly, none of the approaches provides an account of intrinsic teleology and relates it to intentional (creative) agency.

Conclusion and Future Work

We have shown via case-studies that existing CC systems, typically with a focus on human creativity, cannot provide a satisfactory answer to *why* they are being creative because they lack intrinsic goal-ownership. We have adopted the enactive AI framework for a non-anthropocentric account of intentional agency in minimal, embodied agents. Creativity, as commonly perceived, seems to be maximally removed from how simple organisms survive and cope within an ever-changing environment. By showing that constitutively autonomous and adaptive agents must necessarily perform intrinsically valuable and novel actions, we have grounded two of the strongest factors in the attribution of creativity in the essence of the living. It follows that enactive AI's conditions for an intrinsic value function to reflect an agent's own goals are also necessary for intentional *creative* agency. Saunders (2012) notes that AI was missing a means to realise constitutive autonomy. We have referred to operational principles which are hypothesised to also realise adaptivity, and thus laid the foundations for a non-anthropocentric, minimal model of intentional creative agency.

The enactive account to sense-making highlights that we can only assess adaptive creativity in artificial and natural agents with intentional creative agency if we switch perspectives: We have to take *their* embodiment into account, *not ours*. However, it is still to be discussed whether this is subject to the *hard problem* of knowing what it is like to be that system (Nagel, 1974), or if it is sufficient to use introspection in artificial agents to learn about how the system makes sense of its environment as basis for its behaviour. We believe that our non-anthropocentric model can advance progress in CC and extend the scope of the field: if our creativity relies on our embodiment, it is necessarily subject to constraints as in other embodied agents. There might be other forms of creativity in nature, resulting from different constraints, which could benefit us. AI allows us to explore these in simulations even beyond the laws of the physical world. Similar to a famous endeavour in *artificial life* (Shanken, 1998), we encourage looking beyond creativity in nature, and investigate *creativity as it could be*.

We suggest this as a point of departure for a new research direction in CC, addressing intentional creative agency from the bottom up. One of the biggest challenges will be to close the explanatory gap between the creative intentional agency in our minimal agents and humans. As a first step, we have hypothesised that *aspirational*, i.e. consistently adaptive agents will give rise to more complex creative behaviour. To evaluate our model in practice, we have to address the engineering challenge of building a physically embodied agent that can counteract its precarious existence. We agree with Saunders (2012) that more complex forms of creativity require the interaction with other agents, so extending our model to social creativity using insights from enactivism is a promising next step. We also want to refine the sense-making granularity of the current model by drawing on biosemiotic enactivism (De Jesus, 2016). We are fascinated by the following question: If embodied systems establish their own world of meaning relative to their embodiment, what do their creative products and processes look like, and how do they differ from ours? We suggest starting this investigation with minimal, intentionally creative agents and climbing up the "creativity ladder".

Acknowledgments

CG is funded by EPSRC grant EP/L015846/1 (IGGI). CS is funded by Marie Skłodowska-Curie grant 705643. SC is funded by EC FP7 grant 621403 and EPSRC Leadership Fellowship EP/J004049.

References

- Al-Rifaie, M. M., and Bishop, J. M. 2015. Weak and Strong Computational Creativity. In *Computational Creativity Research: Towards Creative Machines*. Atlantis Press. 37–49.
- Allen, M., and Friston, K. J. 2016. From Cognitivism to Autopoiesis: Towards a Computational Framework for the Embodied Mind. *Synthese* 1–24.
- Ay, N.; Bertschinger, N.; Der, R.; Güttler, F.; and Olbrich, E. 2008. Predictive Information and Explorative Behavior of Autonomous Robots. *European Physical Journal B* 63(3):329–339.
- Bishop, J., and Al-Rifaie, M. 2016. Autopoiesis in Creativity and Art. In *Proc. Conf. MOCO*.

- Boden, M. A. 2003. *The Creative Mind: Myths and Mechanisms*. Routledge, 2nd edition.
- Boden, M. A. 2015. Creativity and ALife. *ALife* 21(3):354–365.
- Bown, O. 2012. Generative and Adaptive Creativity. In *Computers and Creativity*. Springer. 361–381.
- Bown, O. 2015. Attributing Creative Agency: Are we doing it right? In *Proc. 6th ICCA*, 17–22.
- Brooks, R. A. 1991. Intelligence Without Representation. *Artificial Intelligence* 47(1-3):139–159.
- Charnley, J.; Pease, A.; and Colton, S. 2012. On the Notion of Framing in Comp. Creativity. In *Proc. 3rd ICCA*, 77–81.
- Colton, S., and Ventura, D. 2014. You Can't Know my Mind: A Festival of Comp. Creativity. In *Proc. 5th ICCA*, 351–354.
- Colton, S. 2012. The Painting Fool. Stories from Building an Automated Painter. In *Computers and Creativity*. Springer. 3–38.
- Cook, M., and Colton, S. 2015. Generating Code For Expressing Simple Preferences: Moving On From Hardcoding And Randomness. In *Proc. 6th ICCA*, 8–16.
- Davis, N.; Hsiao, C.-P.; Popova, Y.; and Magerko, B. 2015. An Enactive Model of Creativity for Computational Collaboration and Co-creation. In *Creativity in the Digital Age*. Springer. 223–243.
- De Jesus, P. 2016. From Enactive Phenomenology to Biosemiotic Enactivism. *Adaptive Behavior* 24(2):130–146.
- Dennett, D. C. 1989. *The intentional stance*. MIT press.
- Di Paolo, E. 2003. Organismically-Inspired Robotics: Homeostatic Adaptation and Teleology Beyond the Closed Sensorimotor Loop. In *Dynamical Systems Approach to Embodiment and Sociality*. Advanced Knowledge Int. 19–42.
- Di Paolo, E. A. 2005. Autopoiesis, Adaptivity, Teleology, Agency. *Phenomenology and the Cognitive Sciences* 4(4):429–452.
- Dreyfus, H. L. 1992. *What Computers Still Can't Do: A Critique of Artificial Reason*. MIT press.
- Dreyfus, H. L. 2007. Why Heideggerian AI Failed and How Fixing it Would Require Making it More Heideggerian. *Philosophical Psychology* 20(2):247–268.
- Friston, K. 2010. The Free-Energy Principle: A Unified Brain Theory? *Nature Reviews Neuroscience* 11(2):127–138.
- Froese, T., and Ziemke, T. 2009. Enactive Artificial Intelligence: Investigating the Systemic Organization of Life and Mind. *Artificial Intelligence* 173(3-4):466–500.
- Gallie, W. B. 1955. Essentially Contested Concepts. *Proc. Aristotelian Society* 56:167–198.
- Gervás, P., and León, C. 2016. Integrating Purpose and Revision into a Computational Model of Literary Generation. In *Creativity and Universality in Language*. Springer. 105–121.
- Gibson, J. J. 1986. The Theory of Affordances. In *The Ecological Approach to Visual Perception*. Routledge. 127–138.
- Gornev, G. P. 1997. The Creativity Question in the Perspective of Autopoietic Systems Theory. *Kybernetes* 26(6/7):738–750.
- Guckelsberger, C., and Salge, C. 2016. Does Empowerment Maximisation Allow for Enactive Artificial Agents? In *Proc. 15th Int. Conf. ALIFE*, 704–711.
- Harnad, S. 1990. The Symbol Grounding Problem. *Physica D* 42(1-3):335–346.
- Haselager, W. F. 2007. Robotics, Philosophy and the Problems of Autonomy. *Pragmatics & Cognition* 13(3):515–523.
- Hoffman, G., and Weinberg, G. 2010. Gesture-based Human-Robot Jazz Improvisation. In *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, 582–587.
- Iba, T. 2010. An Autopoietic Systems Theory for Creativity. *Procedia - Social and Behavioral Sciences* 2(4):6610–6625.
- Johnson, M. 2008. *The Meaning of the Body: Aesthetics of Human Understanding*. Univ. of Chicago Press.
- Jonas, H. 1982. *The Phenomenon of Life*. Univ. of Chicago Press.
- Jordanous, A., and Keller, B. 2016. Modelling Creativity: Identifying Key Components through a Corpus-Based Approach. *PLoS one* 11(10).
- Jordanous, A. 2015. Four Perspectives on Computational Creativity. In *Proc. AISB Symp. Computational Creativity*, 16–22.
- Kant, I. 1995. *Kritik der Urteilskraft (Critique of Judgment)*. Suhrkamp, 1st 1790 edition.
- Kaufman, J. C., and Beghetto, R. A. 2009. Beyond Big and Little: The Four C Model of Creativity. *Rev. Gen. Psych.* 13(1):1.
- Kirton, M. J. 1994. *Adaptors and Innovators: Styles of Creativity and Problem Solving*. Routledge.
- Maturana, H. R., and Varela, F. J. 1987. *The Tree of Knowledge: The Biological Roots of Human Understanding*. Shambhala.
- Menabrea, L. F., and Lovelace, A. 1842. *Sketch of the Analytical Engine Invented by Charles Babbage*. Richard and John Taylor.
- Nagel, T. 1974. What is it like to be a bat? *The Philosophical Review* 83(4):435–450.
- Noë, A. 2004. *Action in Perception*. Cambridge, MA: MIT Press.
- O'Regan, J. K., and Noë, A. 2001. A Sensorimotor Account of Vision and Visual Consciousness. *Behavioral and Brain Sciences* 24(05):939–973.
- Oudeyer, P.-Y., and Kaplan, F. 2008. How Can We Define Intrinsic Motivation? In *Proc. 8th Conf. Epigenetic Robotics*, 93–101.
- Paul, E. S., and Kaufmann, S. B. 2014. Introducing The Philosophy of Creativity. In *The Philosophy of Creativity: New Essays*. Oxford Scholarship Online. 3–16.
- Pfeifer, R.; Iida, F.; and Bongard, J. 2005. New Robotics: Design Principles for Intelligent Systems. *ALife* 11(1-2):99–120.
- Ryan, R., and Deci, E. 2000. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology* 25(1):54–67.
- Saunders, R., and Bown, O. 2015. Computational Social Creativity. *ALife* 21(3):366–378.
- Saunders, R.; Gemeinboeck, P.; Lombard, A.; Bourke, D.; and Kocaballi, B. 2010. Curious Whispers: An Embodied Artificial Creative System. In *Proc. 1st ICCA*, 100–109.
- Saunders, R. 2012. Towards Autonomous Creative Systems: A Computational Approach. *Cog. Comp.* 4(3):216–225.
- Searle, J. R. 1980. Minds, Brains, and Programs. *Behavioral and Brain Sciences* 3(3):417–457.
- Shanken, E. A. 1998. Life as We Know It and / or Life as It Could Be: Epistemology and the Ontology / Ontogeny of Artificial Life. *Leonardo* 31(5):383–388.
- Stewart, J. 2010. Foundational Issues in Enaction as a Paradigm for Cognitive Science. In *Enaction: Toward a New Paradigm for Cognitive Science*. MIT Press. 1–32.
- Thompson, E. 2004. Life and Mind: From Autopoiesis to Neurophenomenology. A Tribute to Francisco Varela. *Phenomenology and the Cognitive Sciences* 3(4):381–398.
- Turing, A. M. 1950. Computing Machinery and Intelligence. *Mind* 59(236):433–460.
- Varela, F. J.; Thompson, E.; and Rosch, E. 1991. *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press.
- Varela, F. J. 1979. *Principles of Biological Autonomy*. Elsevier.
- Varela, F. J. 1984. The Creative Circle. Sketches on the Natural History of Circularity. In *The Invented Reality: Contributions to Constructivism*. WW Norton. 309–325.
- Ventura, D. 2016. Mere Generation: Essential Barometer or Dated Concept? In *Proc. 7th ICCA*, 17–24.
- Weber, A., and Varela, F. J. 2002. Life after Kant: Natural Purposes and the Autopoietic Foundations of Biological Individuality. *Phenomenology and the Cognitive Sciences* 1(2):97–125.
- Wheeler, M. 2005. *Reconstructing the Cognitive World: The Next Step*. MIT Press.

Narrative-inspired Generation of Ambient Music

Sarah Harmon

Department of Computer Science, Bowdoin College
Brunswick, ME 04011 USA
sharmon@bowdoin.edu

Abstract

An author might read other written works to polish their own writing skill, just as a painter might analyze other paintings to hone their own craft. Yet, either might also visit the theatre, listen to a piece of music, or otherwise experience the world outside their particular discipline in search of creative insight. This paper explores one example of how a computational system might rely on what they have learned from analyzing another distinct form of expression to produce creative work. Specifically, the system presented here extracts semantic meaning from an input text and uses this knowledge to generate ambient music. An independent measures experiment was conducted to provide a preliminary assessment of the system and direct future work.

Introduction

Researchers have long established that artificial agents can learn from humans, in addition to each other, to generate creative solutions (Gervás 2001; Boyd, Hushlak, and Jacob 2004; Bisig, Neukom, and Flury 2007; Codognet and Pasquet 2009; al Rifaie, Bishop, and Caines 2012; Bornhofen, Gardeux, and Machizaud 2012; Dubnov and Surges 2014). In contrast, relatively little work has examined how computational systems might analyze artifacts that reside outside their expert domains. While analogical reasoning is well-established as a technique in artificial intelligence, it rarely involves mapping from a source to a genre-distinct target. Even less considered is this type of mapping for the generation of creative works, despite the fact that humans naturally face such tasks daily.

This work provides a step toward what we will term *extra-inspired systems*, i.e., systems that can learn how to usefully and creatively map and/or transform semantic concepts from remote domains to their own, at or beyond the level of human capability (refer to Figure 1). Examples of extra-inspired systems include a poetry generator that can learn from a painting, or a dancing robot that is able to incorporate ideas from the tone or timbre of someone’s voice into its choreography. In both cases, the creative system must gain and apply knowledge from outside its primary domain. In so doing, the connection between the source of inspiration and the resulting product should ideally be apparent

to the audience. This is not to say that creative products could not result if this connection is not made; rather, we assume extra-inspired systems should possess this feature to promote computer-human storytelling across domains.

Here, we introduce a framework that uses semantic information extracted from natural language texts to produce ambient music. This is a challenging task to pursue, given that music composition is complex even for humans (Delgado, Fajardo, and Molina-Solana 2009) and open information extraction is yet a developing field (Gangemi 2013). In light of these barriers, we will discuss how past work provides a foundation for the design and assessment of such an extra-inspired system.

Related Work

Ambient Music Perception and Composition

There are several well-established heuristics for ensuring that music is pleasurable to the human ear. Many of these are rooted in the concept of consistent and natural musical motion. In other words, large leaps in structure or style that occur rapidly are not considered pleasurable. Conjoint melodic motion and efficient voice leading are two examples, reflecting the notion that the ideal path when harmonies change tends to be the shortest possible. Centricity, a quality in which a tonic note is regarded as prominent and serves as the goal of musical motion, is another example of consistency preservation.

Beyond being able to generate pleasing ambient music, it is important that our system engage in creative behavior. Prior work in computational creativity has suggested that creative composition systems should strive to be fully autonomous (Wiggins et al. 2009). Further, the system should be able to generate music that is *novel* and *valuable* for itself and its audience to be considered creative (Boden 2009). Novelty may be *psychological* (new to the composer) or *historical* (new across society for the first time in history).

We look to a key pioneer in ambient music to suggest what is valuable in a generated ambient composition. Brian Eno, who coined the term *ambient music* and contributed heavily to its development as a genre, points to several crucial features that apply to ambient music specifically. According to Eno, ambient music “is intended to induce calm” and “must be as ignorable as it is interesting” (Eno 1978). We will refer

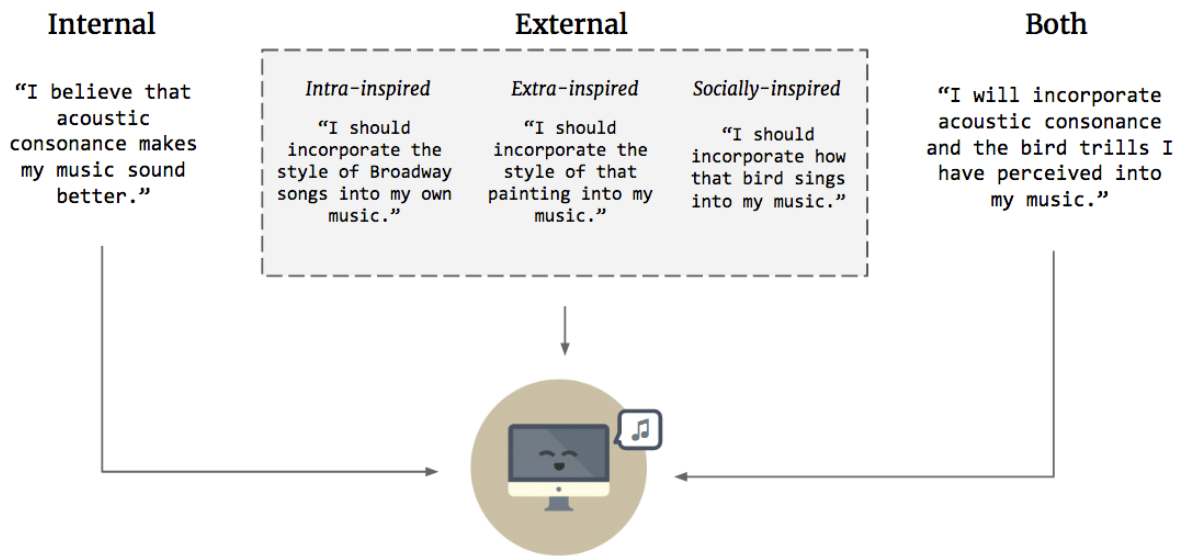


Figure 1: An example of a music-making system is used to illustrate the distinction between systems that are inspired by internal (*internally-inspired*), external (*externally-inspired*), or both internal and external (*ambi-inspired*) cues. These systems may be further categorized by the types of cues to which they attend. For instance, *intra-* and *extra-inspired* systems are defined here as those that apply knowledge from within and outside their expert domains, respectively, whereas *socially-inspired* systems apply knowledge specifically gained through interaction with another being or system. These categories are not necessarily mutually exclusive. Further, note that this categorization does not represent a hierarchy: *ambi-inspired* systems, for example, do not necessarily generate products of higher or lower value than systems that are purely inspired by internal means. The distinction is simply between underlying methodologies.

to these qualities to guide the design of our ambient music generation framework.

Extra-Inspired Music Generation

A number of systems already exist that use visual cues to generate musical experiences (Granito; Pappas; Walker et al. 2007). Mapping text to music is a natural next step, especially considering that homologous brain regions appear to support functions at the level of semantic and temporal structures for music and language in humans (Brown, Martinez, and Parsons 2006). Unlike motion and other visual cues, however, text does not as reliably map to specific changes in sound. For this reason, prior text-to-music systems tend to rely on surface features of text to direct generation rather than semantic cues.

Rangarajan, for instance, recently proposed three strategies for mapping text to music: (1) mapping letters to notes, and their frequencies of occurrence to note duration, (2) mapping only vowels of the words to notes and note duration, and (3) mapping vowels and respective part-of-speech category for each word to notes (2015). This method, however, was acknowledged to be limited because only surface linguistic features were used, in addition to the fact that the resulting music was not necessarily desirable. The support of heuristics such as centrality and efficient voice leading could not be guaranteed.

Davis and Mohammad took text-to-music generation a step further with their proposed *TransPose* system (2014).

TransPose was designed to generate music that captures emotion dynamics in literature (“the change in the distribution of emotion words”). Novels with a more positive emotion profile were represented by assigning a major key to the piece, while novels with more negative emotion words were assigned a minor key. A direct positive correlation was assigned between the frequency of emotion words and tempo of the resulting musical piece.

While these systems are unquestionably important contributions toward meaningful text-to-music generation, much of the substance in the original text does not tend to be preserved in the final product. We chose to couple semantic concept mining with ambient music generation to determine if the interpreted meaning of a text could be clearly and creatively mapped to a novel soundscape. The benefits of pursuing this work include improved responsive audio for interactive playable media, in addition to new, useful, and interesting sound generation for the visually-impaired, as in (Walker et al. 2007).

Perhaps the closest system to the present work is Audio Metaphor (Thorogood, Pasquier, and Eigenfeldt 2012; Thorogood and Pasquier 2013). As in the system described here, Audio Metaphor transforms a natural language query into either (1) a set of audio file recommendations for a soundscape composer or (2) a generated soundscape. The key distinction is the processing of the input text. Audio Metaphor preprocesses the input text by removing common words, and groups the remaining words into a list. This list

serves as the first search query. If the first search query is unsuccessful, more search queries are derived by rearranging or removing some of the words. The input phrase “On a rainy autumn day in Vancouver” would thus result in search queries of *rainy autumn day vancouver*, *rainy autumn day*, *autumn day vancouver*, *rainy autumn*, and so on.

In contrast, the present system performs semantic concept mining on the input text, and can perform transformations on the words or phrases themselves based on its knowledge of the overall story. To illustrate, consider the “on a rainy autumn day in Vancouver” example. In this case, certain properties of the narrative setting are extracted; namely, the fact that it is rainy, autumn, during the day, and in Vancouver. When deriving the search query, the lexical or grammatical categories can be automatically changed (as in *rainy* to *rain*). Generalizations can also be made to make the query more abstract (such as *Vancouver* to *Canada*, *Vancouver* to *city*, or *Vancouver* to *ocean city*). Metaphorical connections may be discovered as well, such as the relation between *rainy* and a particular atmospheric mood. Overall, this method enables the system to represent input as a set of structured concepts rather than a list of unfamiliar words, and to thereby make connections about what is being described. As a result, the system has more techniques at its disposal to make expressive decisions and ensure search query quantity and relevance.

Approach

The Rensa knowledge representation framework was used to encode, extract, and transform semantic information (Harmon 2017).¹ To generate ambient music from text-based narratives, narrative concepts and information regarding how they are related are first gleaned from a provided natural text input (*reading phase*). This knowledge is then automatically translated into search queries for web-based sound libraries (*interpretation phase*), which means that novel sounds may be gathered as the libraries continue to expand. Any returned results are analyzed based on properties specific to ambient music, and combined to generate a set of possible final compositions (*brainstorming phase*). This set is then evaluated by the system based on its understanding of the ambient music domain (*critique phase*). If no results or insufficient results are found during the *interpretation phase*, the system will return to the original text and attempt to extract additional concepts and create new queries. If insufficient concepts are identified in the source text to create a piece of music, the system will still inform the user of any knowledge gained and will suggest sound files based on this knowledge. This enables one to potentially use the system as a computational creativity support tool. The following subsections will explain each major phase of the procedure in more detail.

Read and Interpret

Because state-of-the-art information extraction tools are currently not accurate enough to infer the complete meaning

¹<https://github.com/RensaProject>

of a text (Gangemi 2013), simple concept relation information was extracted using a set of text pattern rules. These rules permit the extraction of facts such as actions and object properties (examples provided in Table 1). When any information is extracted, our system stores where it found the information temporally in the story. It also checks if this information was already known to be true or false.

The Rensa framework also provided several functions for extracting story characters and predicting their gender identity. Figure 2 demonstrates an example of how the present system uses this information when translating extracted semantic information into more abstract search queries.

The search queries were not enclosed in quotes, and only tags and file names were scanned to increase relevancy. The term “-voice” was appended to queries that described properties of entities (such as “cool wind”) to ensure that no distracting speech would be retrieved as part of the search. This term was not appended for queries describing actions, however, as in Figure 2. All query results with at least a 4-star rating were retrieved.

Brainstorm

During the brainstorming phase, the generator selects sounds to be used within each piece. Not all retrieved sounds nor extracted concepts need be invoked when generating a new composition. A primary background sound is chosen among the subset of sounds which are greater than or equal to thirty seconds in length, lending to musical consistency. If no sounds are retrieved with this length, the system still arranges the piece using retrieved sounds, but makes a note of the deficiency. Any other sounds chosen are positioned temporally as in the original source.

To support the generation of a calm, natural-sounding, and ignorable piece, all sounds are analyzed for changes in volume. If it appears that rapid changes in volume occur (*local distraction*), or that a sound is much louder than other sounds in the piece (*global distraction*), the distracting sound or portion of the sound is masked. Small pitch adjustments may also be made depending on the results of using the Krumhansl-Schmuckler key-finding algorithm (Temperley 1999; Shmulevich and Yli-Harja 2000; Zhu and Kankanhalli 2006; Sapp et al. 2011) to support consonance and harmonic consistency.

Critique

The system has several feasible measures for critiquing its own work. Psychological novelty is assessed by examining both the extracted concepts and retrieved sound files used. If the semantic similarity of the extracted concepts is proportionally high to concepts that have been previously extracted (which are stored in memory), then the musical piece is considered less novel. Similarly, if the retrieved sound files already exist in the case library, the system will rank the musical piece as less novel. These criteria enable the system to compare the novelty of its generations relative to each other.

Ignorability and pleasantness are both considered to be qualities that point to how useful the resulting piece will be for the listener. These qualities are assessed by examining drastic changes in dynamics, pitch, and tempo. Changes in

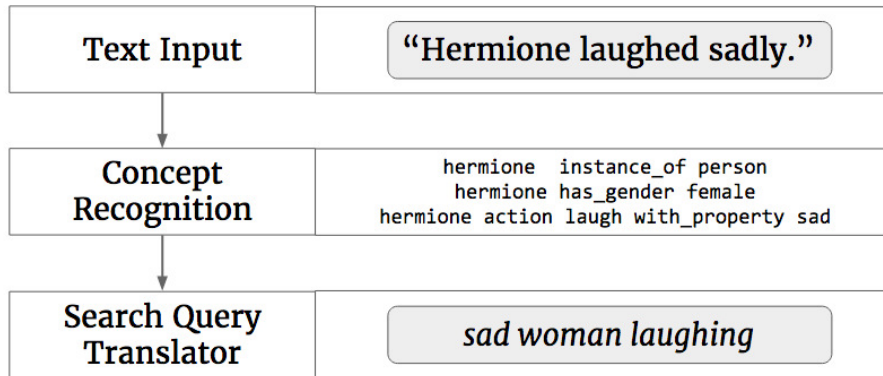


Figure 2: An example that demonstrates how a sentence from an input text is transformed into a search query for a sound file. Semantic concepts are extracted from the text and may be used to make specific terms more abstract. In this case, the word *Hermione* is recognized as an animate actor with a girl’s name, and is abstracted to become the term *woman*. The search query translator can also change the part-of-speech or tense of a word to form a more useful query (e.g., *sadly* to *sad*, or *laughed* to *laughing*).

Rule	Example	Extracted Information
a_1 (RB) VB .	<i>Timothy dreamed.</i>	actor: Timothy , action: dream , tense: past
a_1 (RB) VB (DT) (JJ) NN .	<i>Timothy (sadly) hated (the) (autumn) leaves.</i>	actor: Timothy , action: hate , with_property: sad , action_object: leaves , action_object_property: autumn , tense: past
a_1 (RB) VB a_2	<i>Timothy (really) loved Koko.</i>	actor: Timothy , action: love , with_property: really , action_object: Koko , tense: past
a_1 (RB) VB (PRP\$) (JJ) NN TO a_2	<i>Koko (kindly) gave (her) (helpful) advice to Timothy.</i>	actor: Koko , action: give , with_property: kind , action_object: advice , action_object_property: helpful , action_object_owner: Koko , action_object_recipient: Timothy , tense: past

Table 1: A subset of pattern matching rules used by the present system to extract the occurrence of actions performed by narrative actors (story characters). Here, $a_1, a_2 \in A$, where A is the set of all known actors automatically recognized in the text. The ‘.’ symbol refers to the existence of a full stop or period. All other terms are part-of-speech tags (RB=adverb, VB=verb base form, DT=determiner, JJ=adjective, NN=noun, PRP\$=possessive pronoun, TO=the word ‘to’). Any term that is enclosed within parentheses is considered optional.

dynamics are estimated via a root mean square approach, while tempo is assessed using beats per minute (BPM) detection. Pitch detection is achieved via cepstrum analysis, i.e., fast Fourier transforms (FFTs) coupled with low-pass filtering (Roche 2012; Gerhard 2003). Using this method,

the fundamental frequency is estimated as:

$$\hat{f}_1 = \frac{1}{\tau_{max}}, C(\tau_{max}) = \max_{\tau} C(\tau), \tau > 0 \quad (1)$$

...given that the power cepstrum $C(\tau)$ is obtained by transforming the signal $x(t)$ using a FFT algorithm, con-

verting to the logarithmic scale, and transforming via FFT again:

$$C(\tau) = F^{-1}(\log |F(x(t))|^2) \quad (2)$$

Other qualities to be assessed for a given generated piece include additional measures of consistency, evocativeness of the source text, and atmospheric mood. The number of extracted concepts used helps to assess how evocative a generated musical piece is relative to other compositions. The key of the piece and its components can also be analyzed to assess consistency and predict the emotional mood, similar to (Davis and Mohammad 2014).

A complete evaluation of the automated critique phase is ongoing and will be presented in future work. First, however, we seek to determine whether our system’s procedure is able to generate new and interesting musical pieces while preserving semantic meaning. This will serve to strengthen our understanding of meaningful ambient music generation and build toward improved automated modules for creative reading, interpretation, brainstorming, and critique.

Evaluation

Study 1

Method 140 participants on Mechanical Turk were recruited to read a short (<100 words) passage from *The Wonderful Wizard of Oz* that we will call *Emerald City*. They were then asked to listen to an audio track, and to specify whether the audio reminded them of the passage. Participants were required to provide at least one justification for their answer. The experimental group listened to an audio track which actually corresponded to the passage, while the control listened to a track generated from distinct semantic concepts. Specifically, this track was generated from processing another text passage, which we will refer to as *Dungeon*. Each audio track was limited to 30 seconds, and a fade out effect was applied during the last second of each piece.

Results 45.8% of the experimental group and 11.8% of the control described the audio track as pairing well with the text passage. A two-proportion hypothesis test suggested that the difference between the two groups was statistically significant ($p < 0.05$). Spearman analyses further suggested no statistically significant correlation existed between the text-audio matching decision and (1) whether the participant had a certain gender identity ($\rho = 0.0610$), (2) read the *Emerald City* passage before ($\rho = 0.127$), or (3) was familiar with concepts presented in the passage, such as Dorothy and her friends ($\rho = -0.112$).

Approximately 17% of the experimental group (compared to no participants in the control) explicitly remarked in their explanation that the audio would be suitable as background music for a film adaptation of the passage. Other perceived concepts, such as the emotions (e.g., serenity, awe) that the music conveyed could not reliably predict a participant’s decision.

Study 2

Method To further explore how participants might have arrived at their justifications, personal interviews were conducted as part of a small but more in-depth case study. Six

music tracks were generated from sound files that had been obtained as a result of the present system interpreting six excerpts from narrative fiction. We will refer to these six excerpts and their corresponding tracks as *Emerald City*, *Dungeon*, *Tower*, *Elven Realm*, *Cave*, and *Entering Fantasy World*. Four participants (2M, 2F) listened to the tracks and read the original source texts. All six excerpts were each less than 800 words in length, and all participants were at least moderately familiar with the fictional environments described in the input texts. The tracks themselves ranged from 2-10 minutes in length (2:07 to 9:07).

Participants were then asked to rank the tracks in terms of how well they matched each text, from closest to least closest. They could choose to listen to the tracks again until they were satisfied with their choices. They were also asked to provide a brief explanation for each of their rankings. Afterwards, they were interviewed about the experience. Once the interviews were complete, participants were debriefed and informed of which text each track was meant to convey.

Results Participants justified their rankings by describing the abstract concepts that each musical track brought to mind for them personally. For three of the six text passages (*Tower*, *Dungeon*, *Emerald City*), all concepts identified by participants were exact matches of concepts extracted by the system. For instance, in the *Tower* passage, participants indicated that they were listening for a “roaring fire”, “wind”, and some indication of “stairs”. A subset of the system’s extracted concepts included “roaring fire”, “windy nights”, and “winding mahogany staircase”. In the remaining three tracks, the system identified either only some of the concepts identified by participants, or concepts that were semantically related but not perfect equivalents. As an example, the system extracted “fragrant grass”, “cool wind” and “bright day” from the *Elven Realm* passage rather than the participant’s suggestion of “springtime”.

Five out of six tracks were highly ranked (first or second choice) as evocative of their original source text by at least one participant. However, the majority of participants agreed about which track best represented a passage for only four of the six (*Tower*, *Emerald City*, *Cave*, *Entering Fantasy World*). By far the most difficult track for participants to match correctly to its input text was the *Entering Fantasy World* passage. Interestingly, participants also refrained from naming concrete evocative concepts for this passage, with several choosing instead to listen for an overall ominous sound.

Discussion

In this paper, we presented a system that interprets the semantic content of input texts, and uses this knowledge to manipulate and organize retrieved sound files into novel pieces of music. Preliminary interviews and experimental assessment suggest the system’s ability to successfully identify relevant explicit information in texts is promising. Generally, however, not being able to understand and act on implicit information (such as narrative themes, emotions, and beliefs) is a fault of the current implementation. Another failing is that of disambiguating lexical meaning. As one example,

“house” is a musical genre, although it is likely not to be meant as such in the source texts used here. Similarly, the *Dungeon* was described as an “underground room”, which resulted in the retrieval of a sound clip related to the London Underground. Future implementations should perfect the art of interpreting text, including that of the informal metadata (title, description, tags, etc.) associated with a certain sound file. As research in open information extraction continues to improve, it is expected that our system will demonstrate better performance in reading and interpreting inputs.

In Study 2, participants indicated that the original texts and generated musical pieces were interesting, but also that the greater lengths of each potentially contributed to cognitive load and boredom. Features such as passage length and writing style may thus have influenced how the texts were perceived. *Entering Fantasy World*, for example, was the longest passage (778 words), and this factor may have caused participants to not attend to explicit descriptions in the original text as closely. Overall, participants appeared to describe the passages in more abstract terms when they were longer, which is likely due to memory abstraction (Bransford and Franks 1971).

Convincing sound design, too, is a complex field. Experts recognize that sometimes the most appropriate sound for a target experience is an exaggerated imitation or otherwise mere representation of the real thing (Sonnenschein 2001). We encountered this obstacle with the present system, as some participants in Study 2 unknowingly interpreted certain sounds as distinct from their intended target. For instance, the retrieved sound effect of snowflakes falling in *Entering Fantasy World* was mistaken for a “crackling fire in the background”. Future research should seek to identify factors which influence reader and listener perception within similar contexts.

More broadly, future work should thoroughly categorize the space of extra-inspired systems and investigate how, and to what degree, state-of-the-art creative systems might automatically learn from each other. This learning process should take into account their diverse range of underlying philosophies, key principles (of creative process, system design, etc.), and final products. It is the author’s hope that these next steps will build toward creative systems capable of meaningfully inspiring each other, as well as the rest of the world.

References

- al Rifaie, M. M.; Bishop, J. M.; and Caines, S. 2012. Creativity and autonomy in swarm intelligence systems. *Cognitive Computation* 4(3):320–331.
- Bisig, D.; Neukom, M.; and Flury, J. 2007. Interactive swarm orchestra. In *Proceedings of the Generative Art Conference*. Milano, Italy.
- Boden, M. A. 2009. Computer models of creativity. *AI Magazine* 30(3):23.
- Bornhofen, S.; Gardeux, V.; and Machizaud, A. 2012. From swarm art toward ecosystem art. *International Journal of Swarm Intelligence Research (IJSIR)* 3(3):1–18.
- Boyd, J. E.; Hushlak, G.; and Jacob, C. J. 2004. Swarmart: interactive art from swarm intelligence. In *Proceedings of the 12th annual ACM international conference on Multimedia*, 628–635. ACM.
- Bransford, J. D., and Franks, J. J. 1971. The abstraction of linguistic ideas. *Cognitive Psychology* 2(4):331–350.
- Brown, S.; Martinez, M. J.; and Parsons, L. M. 2006. Music and language side by side in the brain: a PET study of the generation of melodies and sentences. *European Journal of Neuroscience* 23(10):2791–2803.
- Codognot, P., and Pasquet, O. 2009. Swarm intelligence for generative music. In *11th IEEE International Symposium on Multimedia (ISM’09)*, 1–8. IEEE.
- Davis, H., and Mohammad, S. M. 2014. Generating music from literature. *arXiv preprint arXiv:1403.2124*.
- Delgado, M.; Fajardo, W.; and Molina-Solana, M. 2009. Inmamusys: Intelligent multiagent music system. *Expert Systems with Applications* 36(3):4574–4580.
- Dubnov, S., and Surges, G. 2014. Delegating creativity: Use of musical algorithms in machine listening and composition. In *Digital Da Vinci*. Springer. 127–158.
- Eno, B. 1978. Music for airports liner notes. *Music for Airports/Ambient* 1.
- Gangemi, A. 2013. A comparison of knowledge extraction tools for the semantic web. In *Extended Semantic Web Conference*, 351–366. Springer.
- Gerhard, D. 2003. *Pitch extraction and fundamental frequency: History and current techniques*. Regina: Department of Computer Science, University of Regina.
- Gervás, P. 2001. An expert system for the composition of formal Spanish poetry. *Knowledge-Based Systems* 14(3):181–188.
- Granito, G. Generating music through image analysis. *Scientia Review*.
- Harmon, S. 2017. *Narrative Encoding for Computational Reasoning and Adaptation*. Ph.D. Dissertation, University of California, Santa Cruz.
- Pappas, C. Generation of music through images. *Scientia Review*.
- Rangarajan, R. 2015. Generating music from natural language text. In *Digital Information Management (ICDIM), 2015 Tenth International Conference on*, 85–88. IEEE.
- Roche, B. 2012. Frequency detection using the FFT (aka pitch tracking) with source code.
- Sapp, C. S.; Smith, J. O.; Chafe, C.; and Selfridge-Field, E. 2011. *Computational methods for the analysis of musical structure*. Stanford University.
- Shmulevich, I., and Yli-Harja, O. 2000. Localized key finding: Algorithms and applications. *Music Perception: An Interdisciplinary Journal* 17(4):531–544.
- Sonnenschein, D. 2001. *Sound design*. Michael Wiese Productions.
- Temperley, D. 1999. What’s key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception: An Interdisciplinary Journal* 17(1):65–100.

Thorogood, M., and Pasquier, P. 2013. Computationally created soundscapes with audio metaphor. In *Proceedings of the Fourth International Conference on Computational Creativity*, 1.

Thorogood, M.; Pasquier, P.; and Eigenfeldt, A. 2012. Audio metaphor: Audio information retrieval for soundscape composition. *Proc. of the Sound and Music Computing Cong.(SMC)*.

Walker, B. N.; Kim, J.; Pendse, A.; et al. 2007. Musical soundscapes for an accessible aquarium: Bringing dynamic exhibits to the visually impaired. In *ICMC*.

Wiggins, G. A.; Pearce, M. T.; Müllensiefen, D.; et al. 2009. *Computational modeling of music cognition and musical creativity*.

Zhu, Y., and Kankanhalli, M. S. 2006. Precise pitch profile feature extraction from musical audio for key detection. *IEEE Transactions on Multimedia* 8(3):575–584.

The Creative Machine

James Hodson*

Jožef Stefan International Postgraduate School
Jamova 39, 1000 Ljubljana, Slovenia
james.hodson@ijs.si

Abstract

This paper offers a critical review of the underlying assumptions in the field of *Computational Creativity*. We present and integrate the state of the art in the search for machines that could be considered creative by human standards. Through the lens of existing literature, philosophical thought, and empirical experimentation, we propose ways to better understand the roots of creativity, and a new approach for its investigation within the field of Artificial Intelligence.

1 Introduction

What is creativity, and what would make a machine *creative*? This is the question that beguiles the literature in the field of Computational Creativity, oft repeated and cited as too difficult to nail down (see, for example, (?), (?), (?)). The field of Computational Creativity is tasked with both defining the philosophical foundations of the search for creativity, and transferring this tentative understanding into real machines that are convincing and valuable to society. A rather sizeable literature has emerged in recent years that attempts to provide grounding for the field by defining it, in general terms, as “building software that exhibits behavior that would be deemed creative in humans.” (?). This is a functional definition that provides little guidance as to the areas that should be examined in our search for knowledge, and begs the philosophical question of the roots of creative behaviour more generally.

Computational Creativity is not a new field. It is considered a sub-field of Artificial Intelligence, and the thirst to understand how a machine might appear to undertake creative acts akin to humans has lasted at least as long as AI itself. Among the first attempts at defining the requirements for creative acts, Newell, Shaw, and Simon (?) provided the following four heuristics:

1. The answer has novelty and usefulness (either for the individual or society).
2. The answer demands that we reject ideas we had previously accepted.

*The research project leading to these results received funding from the EU Research and Innovation programme Horizon 2020 under grant agreement No. 675044 (BigDataFinance).

3. The answer results from intense motivation and persistence.
4. The answer comes from clarifying a problem that was originally vague.

In what is considered one of the foundational works in the field, Margaret Boden (?) states that creative acts represent (1) *novel combinations* of familiar ideas, (2) *explorations* of the potential of conceptual spaces, and (3) *transformations* that enable the generation of novel ideas. In short, Boden believes creative acts need to be novel, surprising, and valuable.

More recently, the literature appears to tacitly accept that a notion of creativity is difficult to pin down. In their *apology* for the field of Computational Creativity, Colton and Wiggins state:

“[P]erhaps creativity is, for some proponents of AI, the place that one cannot go, as intelligence is for AI opponents. After all, creativity is one of the things that makes us human; we value it greatly, and we guard it jealously.” (?)

Among further refinements to these definitions is the idea that {em creative processes} in varying disciplines should be viewed as “family resemblances” rather than a rigid set of requirements (for an overview of family-resemblances in concept formation from a category-theory perspective, see (?)). We make no claim regarding how specific concepts more or less suggest {em creativity} to observers across disciplines, and fully allow for the contextualisation of these markers.

However, none of these suggestions provides a satisfying definition of what it is to be creative, though each in turn captures certain qualities that may be associated with particular instances of creative processes in one domain or another. In fact, the elephant in the room appears to be that the notion of “creative” is no clearer or more meaningful in our minds than would be “imaginative”, or “playful”. These concepts that we use to personally categorize behaviour ex-post do not have a stable grounding in the processes that generate them. The search for an a priori understanding of an ex-post phenomenon may be a red herring, which would go some way to explaining the case-based example driven approach that researchers in the field have exhibited.

To date, experimentation in the area of Computational Creativity has tended in two primary directions. The first

camp aims to discover the set of domain-independent creative processes, able to generate creative artefacts when applied in any domain. Such processes include *conceptual blending* and *bisociative discovery* of new concepts. The second camp looks to creative processes that are active in specific domains, such as music production, painting, pun generation, and poetry. Rather than find governing principles, the second approach emulates and extends forms specific to each discipline. (?)

In this paper, we aim to provide a more rigid foundation for the exploration of creative behaviour in a computational framework. We will proceed by offering a detailed philosophical discussion of the roots of creativity as perceived and examined by human beings. This discussion will provide a backbone of definitions for further exploration of the kinds of questions that are both useful and interesting to those engaged in scientific research in this area. Our philosophical discussion will be augmented with an empirical exploration of human creativity (and the expectations thereof), through a comprehensive dataset of web searches for creative artefacts and processes. Although not definitive, our empirical exploration offers the first attempt to tie a functional definition to a concrete understanding of this area of study. Finally, we will draw from a natural experiment of creativity by comparing the occurrence of *creative* assertions in human and machine-based reasoning. This test will serve to highlight the areas of investigation noted in our philosophical discussion, allowing us to tie together the structural foundations and future aims.

2 Philosophical Foundations

As with most discussions of a philosophical nature, it is often necessary to start with definitions. In our case, definitions are also the object of our search. As we set out on this path, it is important to make clear that efforts to date under the flag of *Computational Creativity* have value independent of its taxonomical roots. No part of our discussion is aimed at undermining the great progress that has been made in a variety of disciplines, from automating the process of mathematical discovery (?) to visual arts (?), slogan generation (?), musical composition (?), and many more. Our aim in this discussion is to better understand what we mean by *computational creativity*, and how this relates to creativity and creative processes more broadly. Through a renewed understanding of the concepts we are trying to model, and how they fit together with other concepts nearby, we hope to provide a unifying theme for Computational Creativity and Artificial Intelligence that can help guide future research efforts.

2.1 Definitions of Creativity

Our modern understanding of the term *creativity* has been affected by several tides over the course of many hundreds of years. Its essence is the literal “process of making”, stemming from Ancient Greek culture, which had no concept of *creation*, since all arts were seen as following pre-defined rules that strived for the ultimate ideal forms of each concept. In particular, in *The Republic* (see (?), Republic X,

597a-e), Plato asks whether a painter and craftsman should be considered *makers* of things, or simply imitators of the true forms and ultimate ideals. In fact, he goes on to elucidate and agree that only the god could be the *natural maker*, and *all* other makers are in one way or another *imitators* of the forms in nature.

Later, in Roman culture, there were two concepts, “*facere*” and “*creare*”. However, the term “*creare*”, which is the root of “*create*” and “*creativity*” in modern usage, referred strictly to the creation of something out of nothing (*ex-nihilo*) (?). Although poets were also sometimes allowed to partake in acts of creation, the meaning was reserved for things and concepts that did not previously exist, and were not merely combinations of existing artefacts. In fact, the term was most usually used to mean *the divine act of creation*. For the manufacturing of objects, such as tables, chairs, houses, Romans would use the functional term *construct*, whereas the arts had specific words for *painting*, *sculpting* and so on. The concept of creation bore no connection to the activities of daily life, though these were no more or less *creative*, arguably, than today.

In matters of scientific progress, such as the invention of flying machines, computers, and wind turbines, the general belief would have been that these ideas were incrementally “discovered” in nature. Analogously, today we might claim that all scientific discoveries, even the most important of our time, are simply incremental and nuanced shifts in perception atop a mountain of communal technical and scientific knowledge. It is rare that discoveries happen in isolation, indicating some general state of uncovering from pre-existing knowledge and pre-existing mental processes, rather than creation out of nothing.

To this day, the distinction between “making” and “creating” is heavily blurred. Colloquially, English speakers often refer to creating *meetings*, *websites*, *companies*, etc., despite the fact that none of these acts would habitually be considered *creative* in their own right. On the other hand, the determination of the kinds of artefacts considered to be *creative* by society is highly subjective. There exist very few, if any, universally accepted examples of creative behaviours and artefacts, and the individual decision can vary according to several parameters, including:

- Familiarity with the subject matter;
- Cultural norms; and
- Local context.

In the literature on Computational Creativity, there is little resistance to the idea that judgement of creative artefacts is contextual, subjective, and knowledge-dependent (?). In fact, most human endeavours fall prey to these factors—*Intelligence* itself is no more stable in the minds of individuals. However, whereas intelligence has come to refer to the specific cognitive processes that allow an agent to learn from and reason about the information available to them (the manipulation of symbols towards a discriminative objective), creativity possesses no analogous claim. In the AI sub-field of Machine Learning, we can talk about *learnability* through the PAC framework, allowing us to make strong statements

about what can be learned, and how quickly (?). In Computational Creativity, although there have been numerous attempts to specify theoretical frameworks for the purpose of evaluation (see, e.g. (?)), none of them fundamentally captures a concept of creativity from the underlying processes.

A fundamental misunderstanding Earlier, we provided an overview of the most broadly cited characterizations of *creativity*. These, usually vague lists of property attributes that creative actions must conform to, do not capture a universal essence of creativity. However, work in the field of computational creativity often refers to creativity as if it were an absolute phenomenon (as a universally accepted fact), whether absolute globally, or with respect to a particular domain. We dispute such an implicit assumption, both from the perspective of its necessity to the study of computational creativity, and from the perspective of objective reality—there is no need for it, and it does not exist.

Creativity is an ex-post phenomenon,¹ whereas Computational Creativity has tacitly assumed it to be an ex-ante set of cognitive processes.² The same sets of cognitive process that allow agents to reason about the world around them, to survive, and evolve knowledge for the future, already envelope the properties necessary and sufficient for the generation of artefacts that fit our classifiers for “creativity”. Attempting to reverse engineer the infrequent by-product of these processes is problematic, and will not lead to a stable-set characterization. That is, creative acts do not systematically appear through particular and differentiable processes of cognitive systems (including brains). They can be the result of any combinations of the available symbolic and statistical manipulations available. More often than not, the same processes that yield creative artefacts, yield useful-but-not-creative outcomes.

Does this mean that we should give up our search for creativity? Should we let the Artificial Intelligence community at large explore this arena as an indirect consequence of their work? Should we re-brand the field as *Computational Arts and Sciences*?

We should not give up so quickly. But, we must come to terms with the fact that creativity is not separate and special. It is a label applied subjectively to certain societal artefacts, which we might study in two ways. First, by way of analyzing specific classes of artefact considered to be creative, in order to identify the processes of *societal value formation*. This is useful on many levels, and achieving this goal computationally can help improve our understanding of how humans learn, the information used, and the dynamics of “expert” behaviour. Second, by way of harnessing computational models to explore the space of possibilities in the arts and sciences—extending and re-interpreting models of creation while making use of the advantages that Artificial Intelligence techniques can bring (scale, data-driven modeling, reasoning over diverse types of information, avoiding

¹It only exists after-the-fact, by virtue of an observer imbuing an action with such an attribute, independently of the manner of its creation.

²That is, that some cognitive processes are inherently *creative*, and others are not

bias, etc.). The more we help computational models understand the arts and sciences, the more they will surprise us with creative artefacts, but this investigation does not start with the creative artefact as an objective, since this framing is not useful for making broad scientific progress.

In short, by adopting this alternative framing for work in the area of Computational Creativity, we can start to measure progress in a more encompassing manner. Through a better understanding of the processes of value formation, we can better inform future frameworks for evaluation. This re-framing does not invalidate any of the work done to date, but re-poses it in slightly broader and clearer terms, potentially opening up new possibilities for investigation.

On mixtures of cognitive processes When we imagine the vast numbers and types of creative acts that intelligent agents are capable of, it seems logical that these do not stem from particular cognitive processes devoted only to producing creative artefacts. The cognitive abilities that we hold so dear emerge at the intersection of many sub-systems, the repurposing of cognitive pathways, and the particular circumstances/information available. This emergent architecture allows for a richness of generative behaviour that is practically unbounded. In the same way as language is infinitely complex, so too can be the mixtures of creative processes (the grammatical forms) of the mind.

On the question of general vs domain-specific creativity Of ongoing debate in the community is the question whether creativity and creative processes are domain-general or domain-specific (see (?),(?)). That is, are there types of creative processes specific to *musical composition vs. sculpting vs. mathematical discovery*? It should be fairly clear from the discussion above that the distinction is redundant—there are no creative processes. If every creative artefact could conceivably have originated from a different mixture of underlying cognitive processes, then the question of generality and specificity disappears entirely.

On the question of convincing Several authors in the Computational Creativity space bemoan the difficulty of convincing biased observers to attribute creativity to a machine (see, for example (?), (?), (?)), even when those algorithms produce artefacts indistinguishable from those produced by skilled humans. We should note that biases in evaluation performed by humans exist in all corners of human endeavours, from judges in court (?), to evaluations of creative work by different genders (?), and music generation systems (?). In particular, individuals’ relative overconfidence in their own abilities tends to lead to an underestimation of the abilities of others (?), which bias increases as the “other” is more different. It stands to reason that the evaluation of a machine is about as “other” as one might perceive, even in this era of growing habituation to computational devices.

Appropriate evaluation criteria Although a full discussion of the selection of appropriate evaluation criteria for creative systems is beyond the scope of this article, it is clear that this is a problem that beguiles all branches of scientific discovery, and needs to be treated with care. In order

to develop convincing theories, and empirical evidence for or against them, we must be capable of dealing with a variety of human biases, and be wary of the statistical pitfalls of insufficient evidence for the phenomena we claim. More work is necessary towards the understanding of optimal incentive structures, proxy measures, and hypothesis testing in the field of Computational Creativity.

Wiggins (?) notes that an overarching principle for the evaluation of creative acts in machines should be the following:

“The performance of tasks by a computer, which, if performed by a human, would be deemed creative.”

In (?), the authors argue that “creativity is in the eye of the beholder.” Seemingly, the first proposal requires us to define what it means for a human act to be considered creative, and the second tells us that any creative act may only be judged subjectively, individual by individual. Within the framework we are proposing, we want to:

- promote the development of novel techniques in each relevant sub-field;
- reward interesting mixtures of “cognitive processes” used to generate artefacts; and
- encourage the development of systems which can introspect about the processes being implemented.

We believe that a focus on the underlying pillars of intelligence, learning, and reasoning, rather than “creativity” directly, is the best way to maximize the perception of creative value by external observers. As discussed earlier, a deeper understanding of the processes of societal value formation will be the most important precursor to the correct measurement of the ability of machines to imbue those values in the artefacts they produce. The FACE and IDEA frameworks proposed by Colton, Charnley, and Pease (?) are an attempt to pose the question of what a plausible account of computational creativity might look like. The claim that the Computational Creativity Theory (CCT) framework is equivalent to the Computational Learning Theory (CLT) framework for Machine Learning is not convincingly elaborated, but it is fair to say that the FACE model (Framing information, Aesthetic measures, Concepts, and Expressions of concepts) does provide some unifying themes for the discussion of relevant artistic artefacts, and the IDEA model ((I)terative (D)evelopment- (E)xecution-(A)ppreciation cycle) can be seen as covering a type of interaction between such artefacts and their social context/environment. However, by focusing on creativity as the first order goal of the generative act, such models may actually force narrower and narrower explorations of the space, which is generally an undesirable outcome.

2.2 Structure of Investigation

In (?) Pease and Colton provide three main reasons to pursue work in the area of Computational Creativity:

- to provide a computational perspective on human creativity, in order to help us to understand it (cognitive science);
- to enable machines to be creative, in order to enhance our lives in some way (engineering); and

- to produce tools which enhance human creativity (aids for creative individuals).

It is important to note that our clarifying discussion on the nature of creativity, and how this relates to the notion of *computational* creativity, does not require us to deviate from these objectives. In fact, these aims are perhaps more attainable now than ever.

Virtue and Creativity We provide an anecdote from a different area to guide the discussion and investigation of creative acts. Specifically, we will treat the idea of Virtue in the *Virtue Ethicist* normative framework (?). If one is virtuous, then one makes decisions because that is the only way one can imagine behaving—it is an inherent quality. If there is any level of hesitation, questioning, or ulterior motives that drive a decision not to deviate from the virtuous path, then it is not a virtuous act. For instance, a decision not to steal from the local store must be driven by the understanding that it is ethically wrong to do so, rather than because one fears the consequences, or understands the pain it might cause to others, or if the store does not stock the coveted goods. Similarly, the intention to act according to virtuous doctrine cannot be coming from a desire to be recognised as virtuous—one must be good for the sake of being good. Just like silence, when you talk about it, it disappears. Creativity, like virtue, is not taught (?).³ If the aim is to be creative, then the act cannot be deemed creative. Unlike virtue, creativity can only be judged ex-post (virtue requires evaluation of both the process and the outcome), but for the rest, this provides a neat analogy to current philosophical thinking on the matter, from a Virtue Ethics perspective.

Grounding the discussion One may ask whether our motivation of creativity is not at odds with the manner in which it is *taught* in the design professions (such as architecture, marketing, etc.), where people strive directly and explicitly to be creative. Though this may seem like an alluring argument, there are two distinctions to be made. The fact that we strive to teach something does not necessarily mean that it is so. In fact, by teaching our students about creative actions and artefacts, we imbue them with templates that potentially allow them to imitate and develop these themes, but it is unclear which lesson exactly teaches creativity in a pure sense. In addition, although the design professions certainly appear creative, this may be a simple consequence of the design process being iterative and developmental—leading to many poor ideas being dropped, and better ideas validated. This would be well aligned with the argument we have presented thus far. In any case, we direct the interested reader to Plato’s *Protagoras*, for a Socratic exposition of this debate.

We now move in a more practical direction, attempting to find some empirical grounding and suggestive evidence for the statements made so far. Primarily, we will present two empirical investigations of the perception of creativity from both a human and machine perspective.

³The veracity of this statement can be questioned from a variety of angles and philosophical schools of thought, none of which would be helpful to those interested in claiming the existence of a concrete concept: *creativity*

The intention is to motivate two claims:

Firstly, that the common concept of creativity is not as it is often characterized in the literature, that it borrows from and overlaps heavily with other difficult concepts, and that much of the time it is used in ways that have been unexplored by the Computational Creativity field (for good reason). Our claim is that the ambiguity in usage is a consequence of the post-hoc nature of the assignment of *creativity* as a label.

Secondly, that even with fairly simple reasoning infrastructure, machines and humans generate linguistic artefacts of some level of creativity at a non-negligible background rate, independent of the process used. This claim directly supports our view that the same processes that sometimes generate creative artefacts, often (if not usually) generate artefacts not deemed to be creative, and are therefore not relevant to defining creativity.

3 Human Expectations of Creativity

In order to ground our discussion of creative acts, we wish to explore the occurrence of such acts in the wild, to better understand the general expectations, related concepts, and domain specificity of the concept’s usage. Each day, English speakers the world over query the web billions of times, and some portion of these queries naturally encodes the relation between creativity and other concepts. We will leverage these publicly available data sets to infer the types of things associated with creativity in the minds of everyday users of language, and expect to receive an unbiased picture of modern usage.

3.1 Data

We make use of two different sources of web data in order to explore the trends in co-occurrence patterns and concept formation over time. Specifically, we use the Google Trends API (?) and *The Google Web 1T 5-Gram Database* from the Corpus Linguistics Group at FAU Erlangen-Nrnberg.⁴

Google Trends

What is Trends data?

“Trends data is an unbiased sample of our Google search data. Its anonymized (no one is personally identified), categorized (determining the topic for a search query) and aggregated (grouped together). This allows us to measure interest in a particular topic across search, from around the globe, right down to city-level geography.” (Simon Rogers, Data Editor at Google)

The Google Trends API is used to query the frequency of occurrence of concepts across a large cross-section of web searches, allowing the exploration of interest patterns in a particular topic. Topics represent keywords and groups of keywords that are taxonomically disambiguated into classes (e.g. location, organization, person, field of study).

The Google Web 1T 5-Gram Database Google’s 5-gram web language corpus from 2006 provides counts over n-grams (from 1 to 5) of keywords from the public web. The

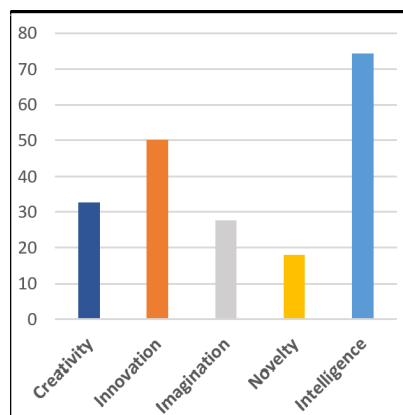
⁴Accessible at <http://corpora.linguistik.uni-erlangen.de/demos/cgi-bin/Web1T5/> as of March 3rd, 2017.

files contain over 1 Trillion tokens, over 95 Billion sentences and from 13 Million unique uni-grams to 1.1 Billion unique 5-grams.⁵ The corpus provides a way to observe keyword co-occurrences across one of the largest cross-domain samples of the English language ever made available. We make use of a web API for exploring the web corpus that has been provided by the Corpus Linguistics Group at FAU Erlangen-Nrnberg. The API allows researchers to choose different statistical measures of importance to rank co-occurrence frequencies among terms of interest.

3.2 Methodology

Web Trends around Creativity In order to explore and compare web search trends relating to creativity and creative processes, we leverage the powerful topic structure available to define a set of structured search terms. “Creative (process)”, “Creative (thing)”, “Creativity(concept)” are combined to provide a baseline. We then search for terms that represent similar concepts to gauge their level of exchangeability through co-occurrence. The additional terms relate to “Imagination”, “Innovation”, “Intelligence”, and “Novelty”. We compare trends for these terms between the beginning of 2008 and the end of 2015. For each concept, we visualize the relative frequency (see Figure 1). For every pair of concepts, we show the correlation between the relevant time series (see Figure 3, Results).

Figure 1: The relative frequency of searches across the 5 concepts of interest: “Creativity”, “Imagination”, “Innovation”, “Intelligence”, and “Novelty”.



Describing Creativity The second of our explorations into the usage of “creativity” in the wild revolves around the large corpus of web text provided by Google, as described above. We want to understand better the sorts of concepts that people associate with creativity, on average. The literature on Computational Creativity is heavily tilted towards the arts, with painting, musical composition, and literature relatively over-represented. In order to make best use

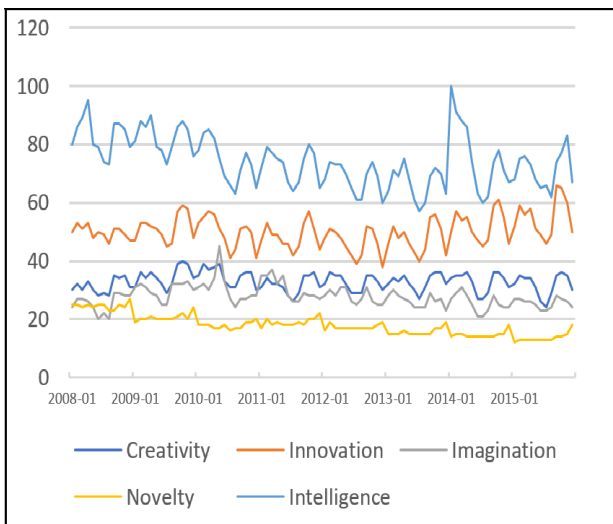
⁵The Google Web 1T 5-gram Database is available from the Linguistic Data Consortium: <https://catalog.ldc.upenn.edu/LDC2006T13> as of March 3rd, 2017.

of the phrasal statistics available in the corpus, we construct a set of search terms that seem to capture the essence of how people might talk about creativity and creative artefacts. Our constructed queries include “creativity”, “creative”, “creatively”, among others. We aggregate the returned co-occurrence frequencies, and rank results following the modified dice coefficient (?). Finally, we strip out obvious noise terms such as “Adobe Creative Cloud” and other proper names. We are left with a set of the most popular terms co-occurring with text about creativity on the public web.

3.3 Results

Our analysis of the trends (Figure 2) shows that “Creativity” is very highly correlated (Figure 3) with “Imagination”, “Innovation”, and “Intelligence”, and that “Novelty” (or “Newness” and “Surprise”, among other concepts) seems not to be correlated with these terms.

Figure 2: The trends of searches across the 5 concepts of interest: “Creativity”, “Imagination”, “Innovation”, “Intelligence”, and “Novelty”.



It is difficult to make strong claims about the absence of correlation in this particular setting, so we focus on the positive associations. In particular, we argue that since these concepts trend together, they likely share many of the same properties. One could certainly imagine similar terminological debates around the foundations of “Computational Imagination” or “Computational Innovation”, were these to become areas of interest to the research community. The results of this particular experiment are merely suggestive, and should not be over-interpreted.

Our analysis of web text data relating to creativity offers interesting insight into the kinds of concepts that people are most interested in when discussing creativity. The most informative terms, ordered by the Modified Dice Coefficient, can be seen in Table 1. The first thing to note is the overlap in terminology from our experiment on search trends among

Figure 3: The correlations between 5 concepts of interest: “Creativity”, “Imagination”, “Innovation”, “Intelligence”, and “Novelty”.

	Innovation	Imagination	Intelligence	Novelty
Creativity	0.70	0.58	0.58	0.03
Innovation		0.25	0.59	-0.12
Imagination			0.38	0.08
Intelligence				0.43

terms. “Innovation and “imagination” are the top two concepts related to creativity, but further down we see “artistic”, “originality”, “inspiration”, “talent”. It certainly seems that these terms correlate with what we have seen in the literature (see for example (?), (?), (?)). It is reassuring that many of the attributes that the various evaluation frameworks look for, are actually everyday relevant concepts for those that speak about creativity. The question of how we measure the contribution of each of these factors effectively still remains, but we leave this discussion open for future elaboration.

Table 1: Collocated keywords with the concept of “Creativity”, by Modified Dice Co-efficient

Collocate	Modified Dice	Frequency	Expected
innovation	26.9433	349035	1673.04
imagination	15.4379	113050	642.4
originality	10.4318	46434	116.84
unleash	7.7493	32807	77.01
ideaflow	6.8198	26233	6.19
artistic	6.3052	48396	706.94
stimulate	5.2166	28843	314.11
ingenuity	4.8916	20775	79.49
bipolar	4.8203	24871	246.49
inspire	3.7247	20865	327.41
latitude	3.6512	27493	680.29
flexibility	3.4415	34976	1162.15
encourages	3.1393	20010	468.76
talent	3.1184	31711	1163.24
enthusiasm	2.9654	17810	401.37

4 Reasoning Creatively

In this section we continue our empirical exploration of creativity in the wild, providing supporting evidence from a very different source, namely one of the largest manually curated ontology projects, Cyc (?). Our aim is to measure the background occurrence of interesting “creative” assertions from a knowledge base that was not designed with any creative output in mind. Once again, this is merely supposed

to add suggestive evidence to our arguments so far, and by no means should it be seen as proof positive for any of the arguments put forward.

4.1 Data

Our set of assertions from Cyc consists of 2,200 sentences generated via the Cyc infrastructure. For each sentence, a random starting node in the ontology is chosen, and an assertion is extracted at random rooted at that node. Since the ontology is the product of individual concepts and rules added manually, along with inference over the graph, many of the assertions that are retrieved follow links that were automatically generated due to logical constraints and reasoning by the Cyc engine. Fortunately, Cyc has a Natural Language Generation model (Logic2NL (?)) that provides human-readable sentences, rather than abstract logical compositions.

Some sample sentences that were generated by this procedure:

- A feeling of anger is likely to be accompanied by a feeling of hatred.
- Every paper contains some weather report.
- In dropping, the "falling" step ends later than the "release of support" step.

4.2 Methodology

For each of the generated 2,200 sentences of natural language, we reviewed them manually according to the novelty, surprise, and value (each on a scale of 1-5) presented by the ideas and concepts evoked. This evaluation loosely follows the criteria suggested by Boden in her seminal work on the subject of Computational Creativity. We chose a cut-off of 12 points to mark sentences as "creative". This choice was based on a ranked review of the quality of output, and can be seen as fairly arbitrary, although sufficient for the point we wish to make.

4.3 Results

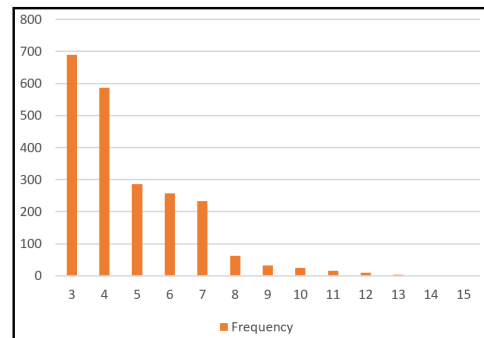
We present results of our exploration, again, as providing ad-hoc suggestive evidence for the arguments that we have presented. We find that the vast majority of the sentences reviewed have little to no trace of a creative foundation. However, in just over 0.5% of instances we do find some sort of noticeable creativity, which does not seem so far away from the background rate of creative artefact generation in our every-day life. Of course, we would need a much more thorough and independent evaluation in order to make any more conclusive statements on the matter. Some of the more noticeable structures include:

- Every resurrection destroys some dead person.
- Haircuts generally affect mammal hair.
- Holding one's breath requires the use of one or more lungs.
- Many actors are charismatic people.
- Middle-class American adults accomplish brushing of teeth easily.

- Monetary values are only monetary values.
- Most auto bodies are taller than most flags.
- No euthanasia is suicide.
- No time lasts longer than forever.
- Sauron The Enemy heads the government of Mordor.

One will notice immediately that novelty and surprise are playing a larger role in the above examples than value, but as an illustrative point some mixture of processes over the Cyc ontology generates somewhat creative artefacts at some rate above zero. See figure 4 for the distribution of scores over the sentences in our corpus.

Figure 4: The frequency of occurrence of each evaluation score for 2,200 Cyc logic2NL generated sentences.



5 Conclusion

In this paper we have proposed a new way in which creative acts might be understood, whether produced by humans, machines, or other intelligent agents. At its core is the argument that creativity is an ex-post phenomenon, and as such the search for the roots of creativity might be a red herring. Instead, we should look for value creation and novelty that emerges from mixtures of all available cognitive processes. None of these will fully characterize what we judge to be creative after the fact, but by broadening our understanding of the driving forces behind creative artefacts we can perhaps start to better understand and model what is going on. Relaxing the assumption that creative processes somehow exist as identifiable entities in the brain needn't change the frameworks for evaluation that are under active development (although it may offer interesting new avenues to explore), nor will it undermine the work that has happened to date in the field. However, by understanding this core distinction we can hopefully contribute even more to our scientific understanding of the cognitive processes involved, and enhance creative output across the board. In addition to a philosophical and theoretical discussion of the nature of creativity, we have presented some suggestive empirical evidence for the concept of creativity in the wild, and for the natural occurrence of creative artefacts even from a stylized setting such as the Cyc ontology. We encourage future work to elaborate these concepts to augment existing evaluation frameworks, and to find new avenues to analyze and expand our understanding of creativity across disciplines.

References

- Ariza, C. 2009. The interrogator as critic: The Turing test and the evaluation of generative music systems. *Computer Music Journal* 33(2):48–70.
- Biles, J. A. 1994. Genjam: A genetic algorithm for generating jazz solos. In *ICMC*, volume 94, 131–137.
- Bluck, R. S. 1961. Plato's "meno". *Phronesis* 94–101.
- Boden, M. A. 1998. Creativity and artificial intelligence. *Artificial Intelligence* 103(1):347–356.
- Bourigault, D.; Jacquemin, C.; and L'Homme, M.-C. 2001. *Recent advances in computational terminology*, volume 2. John Benjamins Publishing.
- Bradeko, L.; Starc, J.; Mladenic, D.; Grobelnik, M.; and Witbrock, M. 2016. Curious cat conversational crowd based and context aware knowledge acquisition chat bot. In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, 239–252.
- Cardoso, A.; Veale, T.; and Wiggins, G. A. 2009. Converging on the divergent: The history (and future) of the international joint workshops in computational creativity. *AI Magazine* 30(3):15.
- Colton, S., and Wiggins, G. A. 2012. Computational creativity: The final frontier? In *Proceedings of the 20th European conference on artificial intelligence*, 21–26. IOS Press.
- Colton, S.; de Mántaras, R. L.; Stock, O.; et al. 2009. Computational creativity: Coming of age. *AI Magazine* 30(3):11–14.
- Colton, S.; Bundy, A.; and Walsh, T. 2000. On the notion of interestingness in automated mathematical discovery. *International Journal of Human-Computer Studies* 53(3):351–375.
- Colton, S.; Charnley, J.; and Pease, A. 2011. Computational creativity theory: The face and idea descriptive models. In *Proceedings of the Second International Conference on Computational Creativity*, 90–95.
- Colton, S. 2008. Creativity versus the perception of creativity in computational systems. In *AAAI spring symposium: creative intelligent systems*, volume 8.
- Colton, S. 2012. The painting fool: Stories from building an automated painter. In *Computers and creativity*. Springer. 3–38.
- Cooper, J. M., and Hutchinson, D. 1997. Plato complete works.
- David, A.; Marianne, B.; and Sendhil, M. 2011. Do judges vary in their treatment of race? *Journal of Legal Studies*.
- Fedyk, A. 2015. Overcoming overconfidence: Teamwork and self-control.
- Google. 2010. *How does Google Trends Work*.
- Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4(3):246–279.
- Lenat, D. B. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38.
- Lenney, E.; Mitchell, L.; and Browning, C. 1983. The effect of clear evaluation criteria on sex bias in judgments of performance. *Psychology of Women Quarterly* 7(4):313–328.
- McMullin, E. 2010. Creation ex nihilo: Early history.
- Neuman, Y., and Nave, O. 2008. A mathematical theory of sign-mediated concept formation. *Applied Mathematics and Computation* 201(1):72–81.
- Newell, A.; Shaw, J. C.; and Simon, H. A. 1959. *The processes of creative thinking*. Rand Corporation Santa Monica, CA.
- Pease, A., and Colton, S. 2011a. Computational creativity theory: Inspirations behind the face and the idea models. In *Proceedings of the Second International Conference on Computational Creativity*. Citeseer.
- Pease, A., and Colton, S. 2011b. On impact and evaluation in computational creativity: A discussion of the Turing test and an alternative proposal. In *Proceedings of the AISB symposium on AI and Philosophy*.
- Plucker, J. A., and Beghetto, R. A. 2004. Why creativity is domain general, why it looks domain specific, and why the distinction does not matter.
- Plucker, J. A. 1998. Beware of simple conclusions: The case for content generality of creativity. *Creativity Research Journal* 11(2):179–182.
- Ritchie, G. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17(1):67–99.
- Sreenivasan, G. 2002. Errors about errors: Virtue theory and trait attribution. *Mind* 111(441):47–68.
- Tomašič, P.; Znidaršič, M.; and Papa, G. 2014. Implementation of a slogan generator. In *Proceedings of 5th International Conference on Computational Creativity, Ljubljana, Slovenia*, volume 301, 340–343.
- Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27(11):1134–1142.
- Ventura, D. A. 2008. A reductio ad absurdum experiment in sufficiency for evaluating (computational) creative systems.
- Wiggins, G. A. 2006. Searching for computational creativity. *New Generation Computing* 24(3):209–222.

Learning to Create Jazz Melodies Using a Product of Experts

Daniel D. Johnson and **Robert M. Keller**

Department of Computer Science
Harvey Mudd College
Claremont, CA 91711 USA
ddjohnson@hmc.edu
keller@hmc.edu

Nicholas Weintraut

Department of Computer Science
Rowan University
Glassboro, New Jersey 08028
weintraun8@students.rowan.edu

Abstract

We describe a neural network architecture designed to learn the musical structure of jazz melodies over chord progressions, then to create new melodies over arbitrary chord progressions from the resulting connectome (representation of neural network structure). Our architecture consists of two sub-networks, the interval expert and the chord expert, each being LSTM (long short-term memory) recurrent networks. These two sub-networks jointly learn to predict a probability distribution over future notes conditioned on past notes in the melody. We describe a training procedure for the network and an implementation as part of the open-source Impro-Visor (Improvisation Advisor) application, and demonstrate our method by providing improvised melodies based on a variety of training sets.

1 Introduction

Substantial work has been done on creation of music by computation, ranging from grammar-based methods (Keller and Morrison, 2007; Gillick, Tang, and Keller, 2010) or genetic algorithms (Biles, 1994) to neural network approaches, including recurrent models (Eck and Schmidhuber, 2002; Franklin, 2004) and deep belief networks (Bickerman et al., 2010). These approaches have had varying success in creating convincing jazz melodies over specific chord progressions.

In the current work, we focus on applying recurrent neural networks to the task of music improvisation. Recurrent models are particularly well suited for sequence prediction tasks, as they are structured to learn patterns across multiple time steps. In particular, LSTM networks (Hochreiter and Schmidhuber, 1997) have been shown to be able to infer complex patterns across many time steps based on data. Additionally, neural network models are interesting to explore because they do not require a large amount of domain knowledge to be explicitly encoded. When given only a limited amount of information about the musical domain, neural networks can use patterns in their training data to discover what makes something musical. This makes them interesting to study as creative systems.

One particularly important component of any neural-network generative model of music is the representation chosen for the task. We use a pair of encodings, motivated

by the observation that good jazz melodies have both interesting contours as well as pitches that are sonorous with the chord progression. One encoding is based on intervals between adjacent notes and the other is based on harmonic relationships with the current chord in the chord progression. Each encoding is processed by a separate network component, and each component produces a candidate note probability distribution. These distributions are then combined using Product-of-Experts (Hinton, 2002) to produce a final distribution over notes that can be trained using the maximum-likelihood criterion.

2 Background

2.1 LSTM Recurrent Networks

Recurrent networks have been shown to be effective at a wide variety of sequence based tasks. In particular, they have been used in the past to model musical data. See Eck and Schmidhuber (2002) and Franklin (2004) for a few examples.

Long Short-Term Memory Networks (LSTM), introduced by Hochreiter and Schmidhuber (1997), have had great success at many sequence modeling tasks. They combat the “vanishing gradient” problem in standard recurrent networks by introducing memory cells that can store state through multiple time steps. An LSTM neuron consists of a series of gates that activate according to the update equations

$$\begin{aligned}f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\C_t &= f_t C_{t-1} + i_t \tilde{C}_t \\o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\h_t &= o_t \tanh(C_t)\end{aligned}$$

where C_t represents the contents of the memory cells at time t , x_t is the input, and h_t is the hidden activations of the LSTM cell. The network consists of one or more layers of LSTM neurons connected with feedback from output to input.

2.2 Product of Experts

Hinton (2002) proposed a system known as Product of Experts (PoE) for combining multiple models of the same

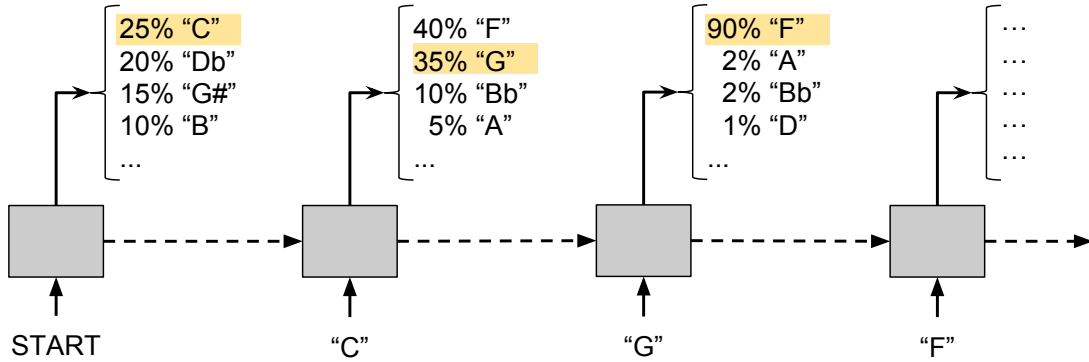


Figure 1: Overview of the network operation, unrolled over four time steps. Inputs are given one time step at a time, and the network outputs a probability distribution predicting the next input. Output is scored based on the probability of correct prediction (highlighted). Dashed arrows represent time-delayed recurrent connections. Although there is also chord input present, we do not show it to avoid clutter.

data. To combine a series of N experts with parameters $\Theta_1, \Theta_2, \dots, \Theta_N$ and associated probability distributions $f_1(\cdot|\Theta_1), f_2(\cdot|\Theta_2), \dots, f_N(\cdot|\Theta_N)$, one can renormalize the product of their individual distributions, i.e.

$$p(x|\Theta_1, \Theta_2, \dots) = \frac{\prod_{m=1}^N f_m(x|\Theta_m)}{\sum_c \prod_{m=1}^N f_m(c|\Theta_m)},$$

where m indexes all experts, and c indexes all possible vectors in the data space. Notice the similarity of this expression to the definition of conditional probability: $p(A|B) = \frac{p(A \cap B)}{p(B)}$. In fact, this product can be interpreted as the conditional probability that all experts choose x , given that all experts choose the same vector from the data space.

When x is a continuous vector in some high-dimensional data space, computing the sum $\sum_c \prod_m f_m(c|\Theta_m)$ and its gradient are both intractable. As such, it is difficult to maximize the log-likelihood of observed data given the model, which led Hinton to propose using contrastive divergence to train such a model. However, if x is a discrete variable in a finite space, as it is here, and each probability distribution f_i is a categorical distribution, the sum can be directly evaluated.

3 Network Structure

Following previous work on LSTM-based models (Eck and Schmidhuber, 2002; Franklin, 2004; Boulanger-Lewandowski, Bengio, and Vincent, 2012), we break each piece into a set of discrete time steps of a specific length (the length of one 32nd-note triplet). The network is designed to receive the previous note played as input at each step, producing as output a set of probabilities for which note to play at the next time step. The network is trained using the maximum-likelihood objective (i.e. choose the network model that assigns the highest probability to the true training data). Figure 1 gives an overview of the network’s operation.

In order to fully develop a LSTM-based model for jazz melodies, we needed to choose a representation for the mu-

sical data for the network to understand. Furthermore, different representations might lead to different types of behavior. For instance, if we simply represented the notes by a bit-vector where each bit corresponded to a single note, the model would not be able to generalize well to different intervals and other patterns, as it would have to learn relationships between each note separately. Franklin (2004) found that representations that were motivated by musical relationships led to better performance than more naive approaches.

Two aspects of jazz melodies on which we decided the network should focus are the *contour* of the melody (i.e. how the notes rise and fall over time) and the *consonance* of the melody with the underlying chord progression (i.e. whether the notes sound good when played over chord or whether they are dissonant). (These aspects could be seen as “viewpoints” in the sense of Conklin and Witten (1995)). We also wanted our network to be *transposition invariant*, as used in Johnson (2017), so as to make similar predictions for inputs that are transposed by focusing on the position of notes relative to the roots of the chords.

To accomplish these goals, we decided to use two encodings, one that focuses on intervals between successive notes, and the other that focuses on intervals of notes relative to the chord progression. We thus split our network into two network modules. The *interval expert* module receives an interval-based encoding, while the *chord expert* receives the chord-based encoding. This enables the network to learn particular relationships relative to the current trajectory of the melody (as learned by the interval expert) and also relative to the current chord progression (as learned by the chord expert). Each of these modules produces a probability distribution over which note to choose at the next time step. This probability distribution represents the level of belief that those notes should be chosen to play next. The encodings used are depicted in Figure 2.

To combine the output of the two experts, we use the product-of-experts equation described in detail in section 2.2. This was motivated by the desire to create melodies

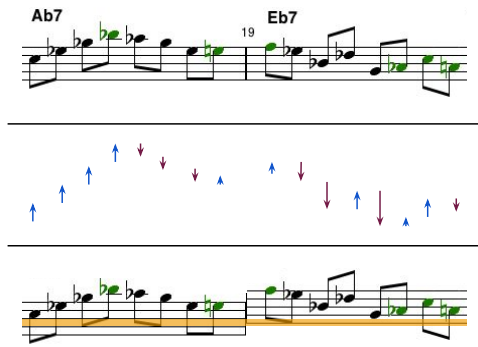


Figure 2: Visualization of the input encodings for the two networks. Top: a section of a training melody. Middle: Interval encoding of the contour. Bottom: Chord-relative encoding of note pitch, with the highlighted space or line representing the root of the chord.

based on both contour and harmonic relationship with the chords. Thus we want to lessen the probability that the combined networks choose a note deemed unlikely by either expert. The combination of the two expert networks is shown in Figure 3.

3.1 Encodings

The interval expert is designed to learn relationships between consecutive notes. Each index in the interval encoding of a note represents a possible interval jump, ranging from -12 (down one octave) to $+12$ (up one octave). This encoding also includes an option of resting, i.e. not playing a note, and one for sustaining the previous note, for a total of 27 possibilities.

The chord expert, on the other hand, is designed to learn relationships between pitches and the chord. This encoding consists of 12 pitch classes, relative to the current chord root (e.g. if the root of the current chord is F, then indices $0, 1, 2, \dots$ correspond to notes $F + 0 = F, F + 1 = Gb, F + 2 = G, \dots$). Again, the encoding also includes an option of resting, and one for sustaining the previous note, for a total of 14 possibilities.

Each expert receives as input:

1. the note chosen at the previous time step, encoded using the corresponding expert's output format (so the interval expert receives the previous interval jump, and the chord expert receives the previous relative pitch class).
2. a *beat vector* giving the position of the current time step in a measure.
3. a *position vector* giving the position of the previous note relative to the upper and lower bounds.
4. a *chord vector* giving the notes of the current chord relative to the expert's current position.

The beat vector for each time step is constructed using a set of reference note durations. Each reference note duration has a corresponding index into the beat vector, and the beat

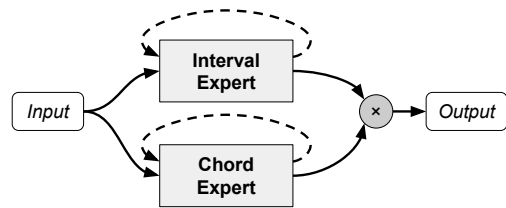


Figure 3: Diagram of the two expert subnetworks. Note that each expert is an independent recurrent network, with output distributions combined using product-of-experts (denoted with \times). Dashed arrows represent time-delayed recurrent connections.

vector is 1 at that index if and only if the current time step is a multiple of that note duration. For instance, using reference note durations of [whole note, half note, quarter note, eighth note], the values of the beat vector at each eighth note in a measure would consist of the following sequence of four-dimensional vectors.

1	1	1	1
0	0	0	1
0	0	1	1
0	0	0	1
0	1	1	1
0	0	0	1
0	0	1	1
0	0	0	1

The beat vector we actually use is of length 9, consisting of whole, half, quarter, eighth, sixteenth, half-triplet, quarter-triplet, eighth-triplet, and sixteenth-triplet.

The position vector consists of two floating-point values. The first element of the vector is 1 at the lowest note of the network range and 0 at the highest note. The second is 1 at the highest note and 0 at the lowest. Each linearly interpolates its values between the two bounds. This allows the network to learn different behavior when playing high notes and low notes.

The chord vector has length 12. Each index of the chord vector corresponds to a pitch class, and the chord vector has a 1 at that index if and only if that pitch class is part of the current chord. These pitch classes are relative to the expert encoding position: for the interval expert, since notes are chosen as jumps relative to the previous note, the chord vector is rotated so that the previous note is at index 0; and for the chord expert, since notes are chosen relative to the root of the chord, the chord vector is rotated so that the root of the chord is at index 0.

Each expert gives probabilities relative to a particular position. To combine the distributions of the experts into a single distribution, we shift the relative distribution encodings to align them to absolute note positions, clip the probability distributions to a specific note range (three octaves), take the product of the distributions, and then normalize the resulting vector back into a categorical probability distribution (i.e. so that the probabilities sum to 1).

4 Training

Our network is trained using the cross-entropy between the network predictions at each time step and the correct notes chosen in the training data. Equivalently, we want to maximize (the log-likelihood of) the probability that the network outputs the training data perfectly, since we want the network to output improvisations that are similar to the training data. We can express the probability of a whole melodic segment as a product of conditional distributions at each time step, i.e. the probability of generating a segment \mathbf{x} from parameters Θ can be factored as

$$p(\mathbf{x}|\Theta) = p(x_0|\Theta)p(x_1|x_0, \Theta)p(x_2|x_0, x_1, \Theta) \cdots \\ = \prod_{t \in T} p(x_t|x_{t-1}, x_{t-2}, \dots, \Theta).$$

Notice that the output of the network at some time step t is conditioned on all previous time steps, due to the recurrent nature of the network. Thus $p(x_t|x_{t-1}, x_{t-2}, \dots, \Theta)$ is given by the output of the network at time t after receiving x_{t-1}, x_{t-2}, \dots as input. To evaluate the full probability of the decoder outputting the input segment, we give the network the observed segment as input, and then accumulate the log-likelihood that the network assigns to the next notes of the observed segment. This gives us the loss

$$L_{\text{reconstruct}} = -\log(p(\mathbf{x}|\Theta)) \\ = -\sum_{t \in T} \log(p(x_t|x_{t-1}, x_{t-2}, \dots, \Theta)).$$

To train our network, we can then compute the gradient of the loss with respect to our parameters Θ by performing backpropagation through time (Werbos, 1990).

In our experiments, each expert subnetwork was implemented as two LSTM layers, each with 300 nodes. During training, dropout of 0.5 was applied to the non-recurrent connections between layers (Srivastava et al., 2014; Pham et al., 2014). We used the ADAM optimizer, introduced by Kingma and Ba (2014), to automatically set the learning rate for our training procedure.

4.1 Generation

After our model is fully trained, we can use it to create new melodies from the learned probability distribution. At each time step, starting from the beginning of the piece, we compute the probability distribution output by our model. We then choose the next note to play proportionally to the probability assigned to that note by our model. This note is then fed back into the model as the input for the next time step, and a probability distribution for the next time step is computed. This process repeats until we have created a full melody segment.

Optionally, we can modify the probability distribution before sampling from it, in order to modify or constrain the output of the network. The first way to modify the distribution is to vary the ratios between note probabilities. Specifically, if the probability of playing note i is given by p_i , we can replace this with

$$p'_i = \frac{p_i^x}{\sum_j p_j^x}.$$

Small values of x make the model more “adventurous” by causing the probabilities to become more similar to each other, and large values of x will cause the model to be “conservative” by sampling high-probability notes more often and low-probability notes less often. Setting $x = 1$ results in the original probability distribution.

Another means of modifying the distribution is to modify the weights assigned to each expert. Instead of computing the probabilities by simply multiplying the probability distributions given by each expert, we can instead use exponents to change the weights assigned to each. If a_i and b_i represent the probabilities assigned by each expert, we obtain

$$p'_i = \frac{a_i^{1+x} b_i^{1-x}}{\sum_j a_j^{1+x} b_j^{1-x}}.$$

This allows the distribution to be biased toward the predictions of a single expert.

Finally, certain elements of the probability distribution can be set to zero in order to prevent sampling particular notes. This can be used either to limit which notes can be played (for instance by only allowing chord tones) or to restrict note durations.

5 Implementation

The software described here is implemented in two parts, one for training and the other for creation. The neural network training program was implemented using the Theano machine learning framework. The program reads and parses input pieces, such as jazz solo transcriptions, in Impro-Visor’s leadsheet notation, trains a neural network model using those pieces as examples, then outputs a “connectome” file, containing the learned parameters of all of the network components for use in the second part.

To explore the use of our neural network model as an educational tool, we implemented the second part in Java as an extension to the open-source Impro-Visor tool (Keller; Johnson). This extension allows a trained connectome file to be loaded and new improvisations to be created in real time over any chord progression, including trading melodies with a human player.

The improvisation part also possesses a number of parameters, which can be used to modify the network output in various ways:

- **Risk Level:** This option determines how the network output probability distribution is scaled, allowing the user to produce either more original but potentially less consonant melodies or more consonant but possibly less interesting melodies.
- **Expert Weighting:** This option allows the user to put more emphasis on the chord expert or the interval expert, allowing the user to tune the relationship between contour and pitch.
- **Rest Limiting:** This option is designed to prevent the network from playing long rests (often due to the presence of long rests in the training set). If enabled, the probability distributions are modified so that rests cannot be played for too many consecutive time steps.

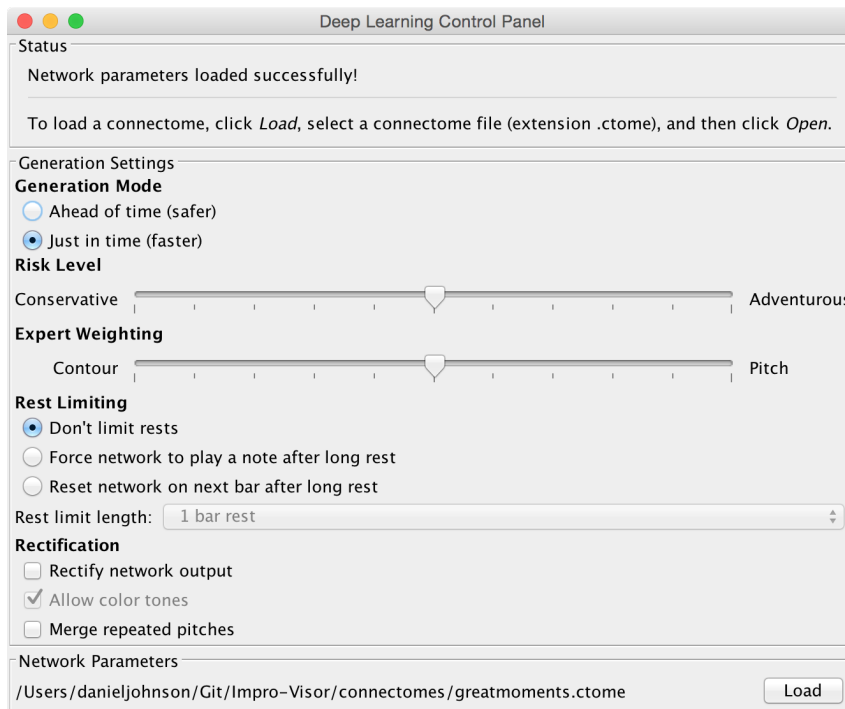


Figure 4: Screen capture of the Impro-Visor deep-learning model control panel.

- **Rectification:** If enabled, this option prevents the network from playing notes that would be dissonant based on the current chord progression. This can help prevent “mistakes” caused by the network randomly choosing dissonant notes with some low but nonzero probability. It can also automatically merge repeated pitches into a single held pitch.

Figure 4 shows the user interface for adjusting these parameters in Impro-Visor.

One advantage of using a recurrent neural network model is that arbitrarily long output can be produced simply by running the network for additional time steps while maintaining the internal state. However, since running the network requires many matrix multiplications, it may take a few seconds to create a few seconds of output. To enable quasi-real-time creation and playback, we implemented a “just-in-time” improvisation process that uses a background thread to produce network output for the next four bars while the previous four bars are being played by Impro-Visor. Then, once playback reaches the end of the existing region, the newly-created four bars are substituted in, and creation begins for the subsequent four bars.

As one application of real-time creation, we integrated the neural network into Impro-Visor’s “passive trading” mode. In this mode, Impro-Visor alternates between playing back newly-improvised music and recording user input, allowing the user and the computer model to synthesize a combined piece. This was a natural fit for the just-in-time neural network improvisation procedure.

6 Results and Evaluation

Using the network model described herein, we successfully trained connectomes from a variety of corpora, ranging from licks in specific keys to full solo transcriptions from a variety of well-known jazz players. In all cases, we are able to create solos of arbitrary length, and in real-time, from these connectomes. Figure 5 compares a melody created with the current model to one created by a grammar trained from the same corpus. Subjectively evaluated by an experienced jazz player (co-author Keller), the improvised solos exhibited occasional shortcomings, such as:

- occasional “wrong” notes, such as a major 3rd that obviously should be minor, or a minor 9th over a minor or major chord, which is usually not advised, but which human players also play on occasion
- occasional rote learning of melody from the original corpus, albeit correctly transposed, also a characteristic of human players

Both of these shortcomings may be partially attributed to the size of our dataset; the first may represent a failure to learn musical rules of thumb, while the second may be indicative of over-fitting. A larger corpus could be used to improve the generalization ability of our model and prevent it from “memorizing” melodies in the training data.

The Impro-Visor website (Keller) gives these examples, along with the Impro-Visor leadsheet files, MIDI files, and the common training corpus used for both the network connectome and the grammar. It should be kept in mind that

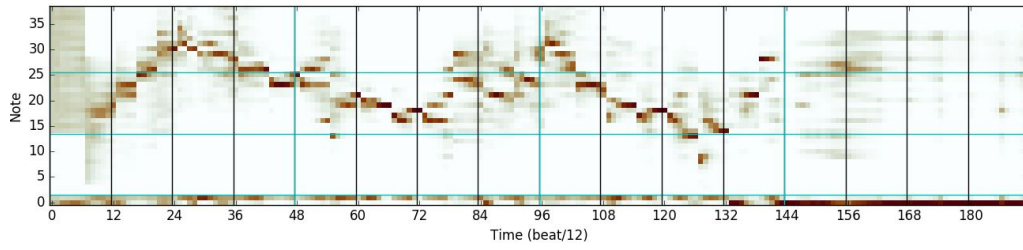
Figure 5: Samples of output created over the chord progression to “Corcovado” by Antonio Carlos Jobim from a connectome trained on a large corpus of licks and solo transcriptions. The corpus did not include a solo for this tune. Top: Version created by our neural network model, with default settings midway between Risky vs. Adventurous, and Contour vs. Pitch preference. Bottom: Version created by Impro-Visor’s grammar-based methods trained on the same corpus. Black notes are notes in the current chord, green notes are “color tones”, i.e. sonorous with the current chord, blue notes are “approach tones” that transition to chord or color tones, and red notes are “outside” notes that do not fall into these categories. To make the comparison fair, rectification, which would generally eliminate red notes, was not used in either sample.

this represents only one connectome and grammar, from one corpus. Several corpora and corresponding connectomes are available with the release, as are many grammars. Each grammar or connectome can be applied to any tune or chord progression.

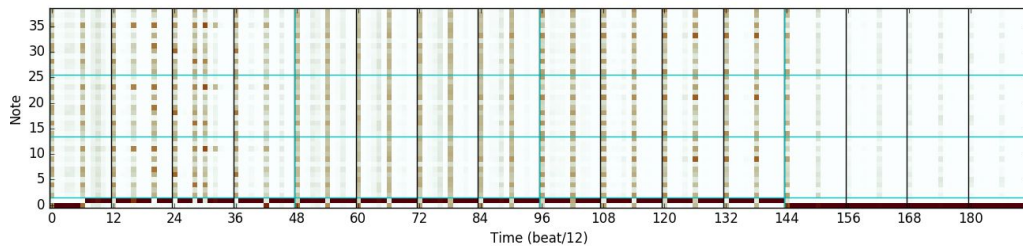
Compared with the grammar-based creation in Impro-Visor, the network-created solos are sometimes less coherent. Our network model was built intentionally to have min-

imal musical knowledge. Since the grammar methods can utilize more prior musical knowledge, and also use Markov chaining of representative melody and abstract melody fragments, the grammar-created samples possess more structure than the network-created samples. Although the current corpus is rather large, an even larger corpus with longer training time might allow our model to learn this latent musical knowledge more effectively, improving the resulting output.

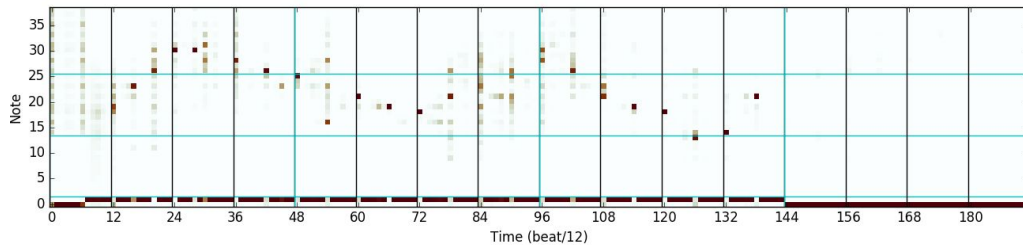
Interval expert



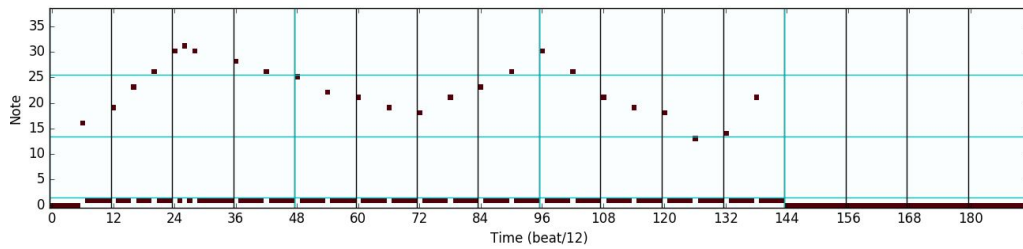
Chord expert



Final probability distribution (product)



Sampled notes



Sampled notes (musical notation)



Figure 6: Output of different components of the trained network while creating a new improvisation. Horizontal lines denote octaves. Vertical lines denote quarter-note durations. The bottom two rows show sustain and rest probabilities. Note that, as desired, the interval expert appears to focus on the direction of movement, the chord expert picks out particular notes that are appropriate, and the combined distribution features note choices that are reasonable to both experts.

Compared with the second author’s earlier work based on deep-belief networks Bickerman et al. (2010), melody creation is much faster with the current model. One reason for this is that deep-belief networks are slow due to internal probabilistic operation. Unfortunately, the learning time for either form of network is on the order of hours, versus minutes for learning grammars.

To attempt to understand what it is that the model is learning, we visually examined the probability distributions generated by each of the experts as well as by the full network. These distributions for a particular improvised four-bar phrase are shown in Figure 6. As predicted, the interval expert seems to learn to make a smooth contour, demonstrated by the focus of the probability distribution around the locations of previously played notes. The chord expert, on the other hand, identifies particular pitch classes as more likely than others. Interestingly, the chord network also seems to have learned to keep track of note durations.

7 Conclusions

We have presented a product-of-experts network approach to learning to create improvised melodies over chord progressions. The network can learn from an arbitrary corpus of existing solos, coded in the form of Impro-Visor’s lead-sheet notation. The learning is then saved in the form of a connectome, which can be loaded into Impro-Visor by the user. Melodies can then be created over arbitrary chord progressions. Comparison with the grammar approach previously available in Impro-Visor indicates a slight subjective superiority of grammars, suggesting that future work might focus on improving the network model to understand larger structural units of melodies.

8 Acknowledgements

We thank the National Science Foundation for providing funding under CISE REU award number 1359170 and Harvey Mudd College for providing the equipment and facilities for this project. We also thank the developers of Theano, which we used to construct all of our models. We appreciate the work of Joshua Zhao in transcribing many examples into the training corpus.

References

Bickerman, G.; Bosley, S.; Swire, P.; and Keller, R. M. 2010. Learning to create jazz melodies using deep belief nets. In *ICCC-X, First International Conference on Computational Creativity*, 228–237.

Biles, J. A. 1994. GenJam: A genetic algorithm for generating jazz solos. In *ICMC*, volume 94, 131–137.

Boulanger-lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 1159–1166.

Conklin, D., and Witten, I. H. 1995. Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24(1):51–73.

Eck, D., and Schmidhuber, J. 2002. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* 103.

Franklin, J. A. 2004. Recurrent neural networks and pitch representations for music tasks. In *FLAIRS Conference*, 33–37.

Gillick, J.; Tang, K.; and Keller, R. M. 2010. Machine learning of jazz grammars. *Computer Music Journal* 34(3):56–66.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8):1771–1800.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Johnson, D. D. LSTMprovisor training program source code. <https://github.com/Impro-Visor/lstmprovisor-python>.

Johnson, D. D. 2017. Generating polyphonic music using tied parallel networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, 128–143. Springer.

Keller, R. M., and Morrison, D. R. 2007. A grammatical approach to automatic improvisation. In *Proceedings, Fourth Sound and Music Conference, Lefkada, Greece, July.*, 330–337.

Keller, R. M. Impro-Visor. <https://www.cs.hmc.edu/~keller/jazz/improvisor/>. Files specific to this paper: <https://www.cs.hmc.edu/~keller/jazz/improvisor/iccc2017/>. Source: <https://github.com/Impro-Visor/>.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pham, V.; Bluche, T.; Kermorvant, C.; and Louradour, J. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, 285–290. IEEE.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Werbos, P. J. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560.

Co-creativity and perceptions of computational agents in co-creativity

Anna Jordanous

School of Computing, University of Kent
Chatham Maritime, Medway, Kent, UK
a.k.jordanous@kent.ac.uk

Abstract

How are computers typically perceived in co-creativity scenarios? And how does this affect how we evaluate computational creativity research systems that use co-creativity? Recent research within computational creativity considers how to attribute creativity to computational agents within co-creative scenarios. Human evaluation forms a key part of such attribution or evaluation of creative contribution. The use of human opinion to evaluate computational creativity, however, runs the risk of being distorted by conscious or subconscious bias. The case study in this paper shows people are significantly less confident at evaluating the creativity of a whole co-creative system involving computational and human participants, compared to the (already tricky) task of evaluating individual creative agents in isolation. To progress co-creativity research, we should combine the use of co-creative computational models with the findings of computational creativity evaluation research into what contributes to software creativity.

Introduction

'This experiment ... has plainly benefited from a lot of human intervention ... To call it "computer-generated" is misleading. "Computer-initiated" and "computer-assisted", though less grabby, are more accurate - and in their own way provide a thought-provoking novelty' (Telegraph)¹

'[Humans] are, essentially, curating and correcting the computers' output' (What's on Stage)²

'... the computer generated claim starts to unravel. There's no software that can put all of these elements together and turn them into a musical. That requires a human' (Engadget UK)³

'I do think the fundamentals [sic] are being nudged by humans' (kevlawdrums on Twitter, Mar 3 2016)

'it's absolutely brilliant - but had a lot of human creative input too' (SemarkJ on Twitter, Feb 27 2016).

¹ <http://www.telegraph.co.uk/theatre/what-to-see/beyond-the-fence-arts-theatre-review-computer-says-so-so/>

² <http://www.whatsonstage.com/london-theatre/reviews/beyond-the-fence-arts-theatre.39847.html>

³ <http://www.engadget.com/2016/03/02/beyond-the-fence-computer-generated-musical/>

These quotes relate to the 'Beyond the Fence' musical theatre project (Colton et al. 2016), in which various computational creativity systems created parts of a musical. Comments on the project included criticism that the project's billing as 'the world's first computer-generated musical' was misleading; it was 'computer-assisted' but not computer-generated. The validity of describing the 'Beyond the Fence' project as an example of *co-creativity* between humans and computers was not explored (Jordanous in press).

In a co-creative scenario involving human and computational participants, how do we attribute and evaluate the creative contributions of different types of participants? In particular, how do we evaluate computational participants relative to human participants, or relative to the system as a whole? Although there have been many advances in computational creativity evaluation tools to date, typically these tools tend to assume we are evaluating a single piece of software, working without creative contribution from other entities. In co-creativity, there are more than one participants contributing to the creative process, but often we cannot delineate the specific contribution of each participant.

As discussed below, Bown (2015) has suggested attributing creative agency to different participants via studying the dynamics of interaction and activities, following Maher's suggestions (Maher 2012) on determining individual and collective creativity. While useful as an approach for how to analyse the contributions of different participants to creativity, Bown acknowledges his proposal is 'despite its long standing in social sciences, quite a radical approach to thinking about attributing creative agency. This view removes the privilege of the human actor' (Bown 2015, p. 21)

Is computational creativity research at a stage where we can afford to remove this 'privilege of the human actor' without consequence, or should we be more cautious in how the contributions of creative systems are attributed in co-creative scenarios? Following Colton (Colton 2008), who closely links the *perceived* creativity of a system to evaluation of creativity, we can rephrase this question:

Is there a problem of bias affecting how well we can recognise computational participants' contributions to co-creative scenarios? If so, how could/should computational creativity researchers address this problem, to make it easier to evaluate and acknowledge computational contributions to co-creativity?

This paper investigates this two-fold question, first considering what problems we should be concerned about in attribution of creativity to computational co-creativity participants. The term co-creativity is established and different manifestations of co-creativity are explored. To understand better what might be needed for an entity to be treated as a potential co-creative participant, co-creativity scenarios between humans and computers are compared to scenarios of collaboration between human participants, and to scenarios involving creativity support tools. Broader questions are also considered, on how to acknowledge computational agents' creativity and how our confidence in attributing creativity is affected by issues of biases. An experimental case study provides data to complement these more theoretical discussions, analysing how people judge a computational system in a co-creative scenario compared to if they were under the impression that this computational system was working in isolation (i.e. not in a co-creative scenario). The paper concludes by discussing the implications for computational co-creativity research; how can we evaluate the creativity of co-creative computational agents, and justify our conclusions - particularly to those not minded towards accepting the possibility of computational creativity?

Background

In a 2005 journal special issue on human-computer creativity, Lubart⁴ provides a commentary on 'How can computers be partners in the creative process' (Lubart 2005). This focus quickly moves, however, to the subtly but distinctly different in meaning: 'focus on ways that computers can contribute to people's creativity'. This de-emphasises the concept of people contributing to computers' creativity, or of people and computers being treated as comparable creative contributors. A 2010 follow-up (Burkhardt and Lubart 2010) shows no particular concessions towards the idea that computers could play a more creative role in co-creativity. To echo Ada Lovelace's words from two centuries earlier, computers are seen as having 'no pretensions whatever to originate anything. It [the Analytical Engine] can do whatever we know how to order it to perform. ... Its province is to assist us in making *available* what we are already acquainted with' (Lovelace 1843) (emphasis in original).

More criticism arose around the possibility of a computational participant to be able to contribute to a set of creative processes in more than a merely supplemental, assistive way. One comment from critics reviewing the recent 'Beyond the Fence' musical theatre project (Colton et al. 2016) was that the project's billing as 'the world's first computer-generated musical' was in fact an inaccurate claim; the project should instead have been described as 'computer-assisted', not 'computer-generated' (Jordanous in press). As discussed elsewhere (Jordanous in press), this was probably a fair criticism that leads us to another useful question: how could computational participants have made a more genuinely co-creative contribution to this project?⁵

⁴This special issue predates by 11 years Lubart's keynote at the 2016 International Conference on Computational Creativity.

⁵This specific question is tackled in Jordanous (in press).

To find evidence to argue for creative computational contributions to co-creativity, we could make some attempt to evaluate the creativity of computational agents using existing or new evaluation methods in computational creativity.⁶ This is problematic, however. Firstly, existing methods are poor for evaluating individual parts of a larger system (Kantosalo, Toivanen, and Toivonen 2015; Jordanous in press), though this has recently been investigated by drawing upon evaluation methods from interaction design (Kantosalo, Toivanen, and Toivonen 2015). Secondly, and the focus of this paper: evaluation of co-creative software can be heavily affected by a deeper question around relative perceptions of creativity in co-creative scenarios between humans and computers. We want to better understand perceptions within co-creativity and make sure we fully capture what evidence is needed to justify claims that the co-creative software participant(s) make valid contributions to co-creative scenarios.

What is co-creativity? Definition

Co-creativity can be modelled in many ways (Candy and Edmonds 2002) but the basic requirement is that more than one participant collaborates actively in a creative process. In computational creativity research on co-creativity, at least one of these participants is computational. Davis (2013) treats co-creativity as a gestalt-like process, where the creative participants contribute in a way which 'the sum is greater than the parts' (rather than a 'division of labor model' where tasks are divided up and allocated to participants in the process (Davis 2013, p. 10). He sees co-creativity as a 'blend' of collaborative improvisation by human and computational agents, 'where a human and computer both make creative contributions to an artwork' (Davis 2013, p. 9). In particular, Davis focuses on a supposed 'new type of human-computer co-creativity' (Davis 2013, p. 9):

'that introduces a computer into this collaborative environment as an equal in the creative process ... in which the human and computer improvise in real time to generate a creative product. Here, creativity emerges through the interaction of both the human and the computer. ... The contributions of human and computer are mutually influential, which is the nature of collaboration and improvisation.' (Davis 2013, p. 10)

This is modelled computationally using an enactive model of cognition (Davis et al. 2015). Further examples of co-creativity in computational models and creative software are starting to emerge (Liapis, Yannakakis, and Togelius 2012; Yannakakis, Liapis, and Alexopoulos 2014; Magerko et al. 2014; Grace and Maher 2014; Kantosalo, Toivanen, and Toivonen 2015; Jacob and Magerko 2015).

Different types of co-creativity

In addressing the question 'How can computers be partners in the creative process' (Lubart 2005), four different roles are identified that computers can take in human-computational co-creativity:

⁶For an overview and discussion of such methods, see the chapters by Ritchie and by Jordanous in the forthcoming 'RADIANCE' Springer volume of Readings in Computational Creativity.

- ‘Computer as nanny’ [p. 366] (helping people to work as efficiently as possible during creative tasks, for example by minimising distractions).
- ‘Computer as penpal’ [p. 367] (an enabler for helping humans to record and communicate their thoughts/ideas).
- ‘Computer as coach’ [p. 367] (‘an expert system, knowledgeable in creativity-relevant techniques’, to help the user to be more creative).
- ‘Computer as colleague’ [p. 368] (‘a real partnership, in which humans and computers work hand in hand’).

Only the fourth of these roles suggests that computers could play anything other than a restricted role in the creative process; however, sadly, even the discussion of ‘a real partnership’ between computers and humans in the creative process is tempered with the assertion that ‘Computers can probably better implement random searches than humans but humans are needed to select the best ideas and perhaps to fine hone these ideas, turning them into viable creative productions’. In other words, computers cannot be creative (according to Lubart) except as an assistant for searching and generating possibilities at random, at which point the human must take over in order for the products of such a partnership to be considered ‘viable creative productions’.

What is necessary in co-creativity for an entity to be recognised as a co-creative participant? Specifically in the context of co-creativity, we can consider how to recognise the creativity of computational participants through analogy. How is creativity of individual participants recognised in co-creativity between human participants? What can we learn from creativity support tools, that are designed specifically to support human creativity? And what methods help us attribute computational contributions to co-creativity?

Comparison with how we treat human collaborators in a creative team comprised fully of human participants

Davis et al. (2015) use human collaboration and improvisation as the basis for their model of human-computer co-creativity. They discuss a ‘*creative trajectory*, which is the shared understanding and intention to make creative contributions in a mutually negotiated and desired direction’ (Davis 2013, p. 11). Co-creativity in general is therefore treated by Davis et al. as a set of meaningful interactions between creative partners, in a call-and-response type model where each partner responds to the other’s communications.

Comparison with creativity support tools (CST) Davis sees human-computational co-creativity as bridging a ‘gap in the CST research literature on how computer colleagues can contribute to the creative process’ (Davis 2013, p. 9). Creativity support tools (Schneiderman 2007) focus on supporting human creativity and disregard computational creativity. CST research tackles questions such as: ‘*What tools, methodologies, and practices can support creativity of individuals in interdisciplinary teams?*’ (Mamykina, Candy, and Edmonds 2002, emphasis in original). It is seen as an HCI research area, not computational creativity/AI (Davis 2013).

Recognising computational agents’ creativity Maher asks the question ‘Who’s Being Creative?’ in co-creative

scenarios with collaborations between different entities (Maher 2012). In these collective creativity scenarios, Maher points out that creative responsibility can be assigned either to individuals within the co-creative scenario or to the collective of entities involved. Maher proposes measuring levels of *ideation* and *interaction* (suggestions overlapping Bown’s later suggestions somewhat (Bown 2015)). Systems can be categorised and ordered along two dimensions of *ideation*, measured along axes that range from modelling to generating, and from the ability to suggest, through enhancement of ideas and to generation. *Interaction* modelling and categorisation is similarly measured in terms of the number of entities involved, human and computational.

Kantasalo et al. argue for a more complex approach: human-computer co-creativity evaluation should be conducted using mixed methods (Kantasalo, Toivanen, and Toivonen 2015), with human opinion a crucial part of analysis. To recognise creativity in an entity, of course, objective systematic evaluation is not necessarily required. Eigenfeldt et al. recognise (at least) five ways of validating the creativity of a system, by considering the perspectives of: the architect of a system; the audience who engage with creative outputs by the system; the academics engaging with publications about a system; domain experts engaging with the system as peers and critics; and the results of controlled system experiments (Eigenfeldt, Burnett, and Pasquier 2012).

Bown’s commentary emphasises the need to recognise the contributions of different agents in the creative process (Bown 2015) and looks at examples of how to do this. He also reminds us that such recognition of creative contribution is necessary in the context of single computational creativity systems: ‘the “islands of creativity” problem’ (Bown 2015, p. 18) highlights the misconception that creativity occurs solely within the bounds of a single creative agent (human or computational). Creative agents engage with the outside world in many ways, as per the ‘Press/Environment’ aspect of the Four Ps of creativity (Jordanous 2016; Rhodes 1961). Creative agents are influenced by external sources, e.g. as audience or peer; and systems influence the world around them by what they are doing. This is captured in creativity models such as the DIFI domain-individual-field-interaction model (Csikszentmihalyi 1988) or Hennessey and Amabile’s (2010) systems take on creativity theory.

A dynamic analysis of creative systems has been offered to analyse creative contributions from different entities (Bown 2015), which defines creativity and the production of output via (a) the interactions between agents and (b) the notion of *becoming*: i.e. how things might be continually dynamically re-created, rather than existing in a stable static form. There is development from Maher’s (2012) proposals to model creative systems via their ideation abilities and interaction. In Bown’s examples of how to attribute creative agency to participants, he outlines how the creative process can be broken down into dynamic activities over a particular timeline, and see how different activities involve or influence different entities. As example, he analyses the creativity of different participants in a musical performance involving two human musicians and various algorithmic performers. The collaboration is broken down into steps, helping to

identify where participants influence each other. (Here the system designer is also considered a participant).

In situations where bias is carefully controlled, Bown's analysis provides detailed qualitative descriptions of creative contributions by the participants in the described scenarios. Sadly, such situations are rare in practice; the approach passes over the problems of bias (conscious or subconscious) that arise in evaluation of computational creativity when computational participants' creativity is examined. The next section investigates these problems in more detail, focusing on the context of co-creativity.

Bias and confidence in recognising creativity

To what extent do bias and differing levels of confidence influence research processes (particularly evaluation) in computational creativity? In Davis's work on co-creativity, it is interesting to contrast (1) his repeated emphasis on the equality of the partnership between human and computer in co-creativity with (2) occasional deviations away from this viewpoint in his writing, towards a view of the computer as the more subservient partner in this process, supporting human creativity without necessarily recognising any creativity on the part of the computational agent. For example (emphases added): 'The proposed system, *Coco Sketch*, encodes some rudimentary stylistic rules of abstract sketching and music theory to contribute *supplemental* lines and music while the user sketches.' (Davis 2013, p. 9). Or: 'Creative computers could understand and work alongside humans in a new hybrid form of human-computer co-creativity that could inspire, motivate, and perhaps even each creativity to *human users* through collaboration.' (Davis et al. 2015, p. 109) Evaluation in (Davis 2013) focuses on discussing/reporting and measuring the creativity of the human participants in a human-computer co-creative scenario, with the only evaluation of the computational participant's contribution being the use of Amabile's Consensual Assessment Technique to evaluate the creative output.

There is a place for soliciting human opinion in creativity evaluation, not least as a simple way to consider the system's creativity in terms of those creative aspects which are overly complex to define empirically, or which are most sensitive to time and current societal context. Recognition of computational creativity in general is however affected by a number of different issues and challenges, many of which are matters of significant, long-standing and/or continuous debate both in the computational creativity community and in research on human creativity. The perception of a computer system as being creative (or potentially creative) is a different issue to the question of whether a computer system actually is creative (Colton 2008). Both of these issues are relevant in this current description of the recognition of the creativity of computational co-creativity participants, in particular the first. Creativity has been described as being 'in the eye of the beholder' (Cardoso, Veale, and Wiggins 2009, p. 17). As emphasised in the Four Ps approach to creativity (Jordanous 2016), the opinions of the audience play a key part in making, distributing and maintaining creativity judgements. Overcoming negative preconceptions about computational creativity can therefore be a necessary hurdle

for computational creativity researchers to negotiate.

Practically it can be tricky to use human judges for evaluating creativity of a computational system. Human evaluators can make a tacit judgement on whether they think something is creative but may find their decision difficult to explain or justify (as investigated in Jordanous (2012b)). In a case study asking people to assess the creativity of systems (Jordanous 2012b), several participants requested a definition of creativity. This study also showed the variance in human opinion; what some found creative, others did not. While larger studies might capture consensus of opinion if it exists, consensus is not necessarily a guaranteed result (Jordanous 2012a; 2012b).

People's competence in evaluating computational creativity can be questioned (Lamb, Brown, and Clarke 2015); evaluations can be influenced by preconceived notions and beliefs. This perception has been discussed from several different perspectives e.g. (Minsky 1982; Moffat and Kelly 2006; Colton 2008; Nake 2009; Reffin-Smith 2010; Pease and Colton 2011; Jordanous 2012b). People may be reluctant to accept the concept of computers being creative, either through conscious reticence or subconscious bias (Moffat and Kelly 2006). Researchers keen to embrace computational creativity may be positively influenced to assign a computational system more credit for creativity than it perhaps deserves. In short, our ability to evaluate creative systems objectively can be significantly affected once we know (or suspect) we are evaluating a computer rather than a human. *Perhaps we evaluate systems differently to how we evaluate people - and perhaps systems' collaboration with people distorts how we attribute creativity to that system?*

Experimental work: an evaluative case study

The above questions probe how we recognise the creativity of computational participants in co-creative scenarios, to assist us in computational creativity evaluation. It would be useful to have data to make comparative judgements of systems' creativity in a co-creative context compared to outside this context. In an ideal world, we could simply ask people directly to compare the creativity of a system in a co-creative context to its creativity when working on its own. Life is not that simple, though; it is difficult to ask directly without encountering subconscious bias or confusion over what it means for computers to be creative (Jordanous 2012b).

Instead, in this case study, the aim is to collect a rough consensus of opinion on how creative a system is: either with the participants under the impression that the system is operating independently on its own; or being informed that the system is operating in a co-creative scenario with a human user. This study investigates the two-part hypothesis:

- i. People will vary in their evaluation of how creative a system is considered to be, depending on the evaluator's impression of whether it is operating on its own or whether it is operating in collaboration with human(s).*
- ii. People are more confident attributing creativity to computational systems operating in isolation, than attributing creativity to computational systems that operate within a co-creative scenario.*

Such comparisons let us investigate if people consider a co-creative system differently if treated as part of a complete co-creative collaboration of human plus computational participant (instead of evaluating the computational participant in isolation from the human collaborator). The null hypothesis, hence, is that there is no significant difference in evaluation or in confidence of evaluation of a creative system on whether it is co-creative or not, and regardless of if an entire co-creative scenario is being evaluated or whether we are evaluating individual participants within that scenario.

We compare the evaluations (and participants' reported confidence in evaluations) of the same system with people under different impressions as to how much it collaborated with other people, and compare the evaluations of the system in isolation to the system as part of a bigger co-creative system of collaboration. We also compare confidence levels for evaluations of a co-creative system (either treating the 'system' (1) as all participants, human and computational, or (2) just evaluating the computational participant), against confidence in evaluating a computational system that works in isolation, with no human-computer interaction.

Method

Three groups of participants were used for this study, each with a minimum of 30 participants. Participants were asked to indicate their opinions on how creative a co-creative computer system is, and were also asked to indicate how confident they were about their answer. The participants were given a brief description of how the system works (using non-technical language and avoiding jargon). They were also provided with sample outputs of the system.

The co-creative system used in this study is the Impro-Visor (Keller 2012) musical improvisation system, which works in conjunction with human student users to generate jazz music in the style of improvised solos. Impro-Visor is intended as a tool to help people develop their ability to construct jazz solos. It learns musical grammars from corpora of jazz music and uses those as generative grammars to suggest solos which can be developed by the human user.⁷

The three groups were divided according to how they were asked to approach this task:

- Group 1: Treat Impro-Visor as part of a co-creative system, evaluated separately from the human user participating in the co-creative process
- Group 2: Treat Impro-Visor + human user, considering both as parts of a human-computer co-creative system
- Group 3: Treat Impro-Visor as a standalone creative system, *not* a co-creative system

Group 1 evaluated Impro-Visor specifically as the computational participant in a co-creative scenario, i.e. considering the software participant's creativity. Group two evaluated how creative the co-creative system was as a whole, i.e. Impro-Visor and the human user. Data for both groups 1 and 2 were collected during lectures on computational creativity evaluation, to students at undergraduate or MSc level. The

⁷Although Impro-Visor recently gained functionality for collaborative real-time improvisation with a user (Keller et al. 2012), this functionality was not included in the evaluation case study.

exercises were completed at the start of these lectures, before the students had been given any information from the lecturer on computational creativity evaluation; the teaching purpose of the exercise was to get students to think practically about how to evaluate creative systems, and prompt them to consider to themselves the issues that arise, prior to any discussions of this in the teaching session.

The third group's data was collected slightly differently. In fact, this study was originally inspired by a previous - and aborted - data collection for an evaluative case study involving Impro-Visor, conducted under a false impression that the output samples used for the case study were indeed generated purely by the system in question working independently. Later it was discovered (personal communications with Bob Keller, 2012), that the system outputs were the result of the system and a human user in collaboration. This was unfortunate for the original study, but a serendipitous inspiration and source of data for this study. This original data on evaluation of Impro-Visor seemingly as a standalone system, not collaborating with a human user but operating autonomously, was collected via an online survey for (Jordanous 2012b) that presented participants with descriptions of four musical improvisation systems and their outputs. Participants were asked how creative they thought each system was, and to give their confidence in their decision.

Groups 1 and 2 evaluated Impro-Visor as a second task after performing a similar evaluation of a non-co-creative system Tale-Spin (Meehan 1976). The main purpose of the Tale-Spin evaluation was mostly to give the participants a practice evaluation task. Though it also provides data to compare the confidence of participants in groups 1 and 2 at evaluating a co-creative system compared to a non-co-creative system, such analysis is limited; many factors influence why we may be more confident evaluating one system than another (as shown by comments made in the study).

Of course the evaluation of one system compared to another unrelated system in a different domain does not give us any sort of complete picture as to how people treat systems in co-creative scenarios compared to non-collaborative scenarios. While the data from evaluation of Tale-Spin is interesting as some sort of yardstick, the evaluation task for Tale-Spin is essentially a practice exercise before the main exercise we are interested in collecting data for.

For the evaluative exercise, the participants were asked: "How creative do you think the XXX system is?" (substituting for XXX the system perspective they were being asked to consider, based on whether they were in group 1, 2 or 3). Ratings of creativity were collected using a five-point labelled Likert scale 0-4: [0] Not at all creative, [1] A little creative but not very, [2] Quite creative, [3] Very creative, [4] Completely creative. Then participants were asked: "How confident are you about the answer you just gave for XXX system's creativity?" (substituting for XXX the system perspective they were being asked to consider). Ratings of confidence were also collected using a five-point labelled Likert scale 0-4: [0] Very unconfident, [1] Unconfident, [2] Neutral, [3] Confident, [4] Very confident.

Table 1: Data on ratings of Impro-Visor creativity on the 0-4 scale, from ‘Not at all creative’ to ‘Completely creative’

Group	1	2	3
N	34	50	120
mean	1.91	1.93	1.88
median	2.00	2.00	2.00
mode	2.00	2.00	2.00
std dev	0.723	0.874	0.913

Results of case study

A summary of the collected data is given in Tables 1 and 2. A two-tailed t-test was used to compare whether there were significant differences in ratings in confidence between the three groups, with significance set to $p \leq 0.05$.

The first set of tests tested the null hypothesis that there was no significant difference in ratings for Impro-Visor’s creativity, across the three groups (i.e. first comparing group 1 and group 2, then comparing group 1 and group 3, then finally comparing group 2 and group 3). No significant evidence was found to disprove the null hypothesis; in other words none of the three different perspectives taken on Impro-Visor significantly affected how creative the system was perceived to be. Across each of the three groups, the Impro-Visor system was rated at between 1.88 mean (group 3) and 1.93 (group 2). This implies a rating of between 2 (‘Quite creative’) and 1 (‘A little creative but not very’).

The second set of tests tested the null hypothesis that there was no significant difference in confidence for rating Impro-Visor’s creativity between pairwise combinations of the three groups. The alternative hypothesis is that a significant difference in confidence has been found.

In comparing groups 1 and 3, i.e. comparing the confidence in rating the Impro-Visor software individually as a co-creative participant and as a standalone piece of software that does not collaborate with a human, a p value of 0.06 meant that although close to being significant, there is no statistical evidence for a significant difference in confidence. In the other two pairwise combinations, however, significant evidence was found to reject the null hypothesis.

Comparing groups 2 and 3, i.e. considering the collective creativity of Impro-Visor and the human user compared to the creativity of Impro-Visor as a standalone program, a p-value of 0.00002 indicated strongly significant evidence of a difference in confidence levels. Remembering that a confidence rating of 3 represents an answer: ‘Confident’, and 2 represents an answer: ‘Neutral’, a mean of 2.74 for confidence in ratings for group 3 contrasts against a mean of 2.02 for group 2. **In other words, participants were significantly more confident at rating the creativity of Impro-Visor if they were under the false impression that it was operating on its own, without user collaboration - compared to trying to rate the collective creativity of Impro-Visor and the human user together.**

Participants were also significantly lower in confidence ($p=0.038$) at completing the latter task compared to the confidence levels showed by group 1 participants, who

Table 2: Data on confidence of participants in providing ratings of Impro-Visor creativity. On the 0-4 scale, this ranges from ‘Very unconfident’ to ‘Very confident’

Group	1	2	3
N	34	50	120
mean	2.44	2.02	2.74
median	2.50	2.00	3.00
mode	3.00	3.00	3.00
sd	0.801	1.008	0.912

had to rate the creativity of Impro-Visor as an individual participant in this co-creativity scenario.

Using t-tests, comparisons were also made between the confidence of participants rating Impro-Visor’s creativity compared to their confidence in rating Tale-Spin’s creativity, for participants in groups 1 and 2 who evaluated both systems. However no significant evidence was found to show that participants felt more or less confident rating the co-creative system Impro-Visor, compared to the standalone system Tale-Spin. This was as expected, due to the many factors that influence confidence in evaluating systems based on differences in processes, products and domain.

Discussion

Regarding the hypothesis under investigation The data shows that while there seems to be little significant difference in the level of creativity attributed to Impro-Visor, people typically felt significantly less confident about evaluating the creativity of a co-creative system of human and computer participants, compared to evaluating the creativity of the computational participant, or if they were under the impression that the co-creative system was actually working as a standalone system with no human collaborator.

One explanation of the data may be that the participants would generally feel less confident about assigning creativity to a collaborative group of participants, compared to attributing creativity to individual participants within that creative scenario. To understand retrospectively how participants decided on their rankings, we can look at the qualitative data collected during the case study described above.

Many participants took advantage of the option to add qualitative comments to support their evaluation data. In the group 3 evaluation scenario, though many comments were given on the musicality of the system, only one participant from the 120 participants involved made a comment about their confidence in their answer:

‘I liked this one better than the other ones, but am really struggling to distinguish between “like” or “approve” and “think it’s creative”.’ [User rated Impro-Visor as 3 for creativity and 3 for confidence]

In groups 1 and 2, more participants commented on their confidence in performing these evaluative tasks. For those in group 2, where participants were evaluating the collective co-creative system of software plus human, no comments at all were made about any difficulties in attributing creativity collectively to a pair of creative collaborators rather than

to the individual participants involved. Of the comments that were made, the following are interesting for showing people's attitudes to the computational participant in the co-creative scenario [their ratings for creativity of Impro-Visor and their indicated confidence are given in brackets]:

"The system has the student to help out on creativity."
[creativity=1, confidence=3]

"These systems can't evaluate their own creativity. The main difference is the human input" [creativity=1, confidence=2]

In group 1, where the participants were typically lower in confidence than the other two groups in their evaluations, evaluating the collective co-creative system of Impro-Visor plus its human user, similar indications were still given as to the relative creative contributions of each participant. One participant underlined the reference to the human user in their answer of how creative the system of 'computer+human' was, presumably to indicate that their attribution of a creativity rating of 2 was because of the human participant (given with a confidence rating of 2). Other interesting comments made by this group's participants included:

"The music sounds like it could have been made by a human alone whereas the story seemed computer generated rather than human written." [creativity=3, confidence=3]

"Quite creative based on the human influence." [creativity=2, confidence=1]

"human element helps with the confidence and knowing more about the workings." [creativity=3, confidence=3]

"If I take into account the fact that a computer was involved I tend to (rather silly and unrealistically) consider the whole system less creative. Bias towards human creativity!"
[creativity=2, confidence=1]

"Impro-Visor: Don't know how much input came from the musician (pre-knowledge?) and how much came from the system?" [creativity=2, confidence=1]

Despite these comments, the data on creativity ratings does not show that people gave *significantly* higher ratings when considering the human as a part of the co-creative system being evaluated (group 2, with a mean of 1.93 for creativity rating compared to 1.91 for group 1 and 1.88 for group 3). Those who added extra comments, however, wished to indicate their reliance on the human part of this co-creative system. Unsurprisingly, perhaps, no participants indicated the opposite sentiment that the computational element helped them feel more confident about their evaluation than if evaluating only the human participant.

Conclusions and implications for future work

How are computers typically perceived in co-creativity scenarios? The study reported here supports the literature review suggestions that participants are less confident attributing creativity in collective co-creativity scenarios including computational participants, arguably because of a reluctance to assign creative agency to computational participants.

The data from the case study in this paper shows that while there seems to be little significant difference in the level of creativity attributed to Impro-Visor, people typically felt significantly less confident about evaluating the creativity of a co-creative system of human and computer participants, compared to evaluating the creativity of the computa-

tional participant, or if they were under the impression that the co-creative system was actually working as a standalone system with no human collaborator. People typically felt 'Neutral' in their confidence of judging the collective creativity of a human-computer collaboration, compared to half way between 'Neutral' and 'Confident' for rating the system individually in this co-creative scenario, and three-quarters of the way from 'Neutral' to 'Confident' for participants' average confidence in rating Impro-Visor's creativity if they had thought the system was working autonomously.

Bown has suggested a working model for how to attribute creative agency to different participants in human-computer co-creativity, based on interactions and dynamic tracing of influence in the creative activities (Bown 2015). This builds on previous suggestions to model co-creative systems individually or collectively via measuring ideation and social interactions in the systems (Maher 2012) and mixed methods (Kantosalo, Toivanen, and Toivonen 2015). There is a certain level of naivety to these approaches - understandable given the desire to move forwards in a more objective and methodical approach for attributing creative agency in co-creativity. With these approaches, we have a decent theoretical starting point; but nonetheless further attention needs to be paid to practical issues that arise in their application.

Human evaluation should indeed be used to recognise and assess the contribution of computational participants in human-computer co-creativity (Kantosalo, Toivanen, and Toivonen 2015). What, though, needs to be done for computers to be perceived as genuine partners in a co-creative process, making a *creative* contribution? How can we demonstrate that in creative scenarios, computational software is not merely limited to the remit of creativity support tools, supporting human creativity, but can become (and exceed the requirements of) a creative 'colleague' (Lubart 2005)? Can we even aim for a point at which human creativity can be seen to support computational creativity?

These are not simple questions to answer, but they should not be passed over. With computational creativity models and software, we can, and have, explored such questions further, to advance understanding of co-creativity more generally. As discussed above, ongoing evaluative research on what makes software appear creative (or what makes software actually creative?) helps us pursue these questions. Once we better understand how to deal with inherent conscious/subconscious biases involved in human evaluation of computational creativity, our working approaches for creativity attribution become more useful as a basis for accurately recognising creative agency in co-creative software.

References

- Bown, O. 2015. Attributing creative agency: Are we doing it right? In *Proceedings of the 6th International Conference on Computational Creativity*. Park City, UT: ACC.
- Burkhardt, J.-M., and Lubart, T. 2010. Creativity in the age of emerging technology: Some issues and perspectives in 2010. *Creativity and Innovation management* 19(2):160-166.
- Candy, L., and Edmonds, E. 2002. Modeling co-creativity in art and technology. In *Proceedings of the 4th International Conference on Creativity and Cognition*, 134-141. Loughborough, UK: ACM.

- Cardoso, A.; Veale, T.; and Wiggins, G. A. 2009. Converging on the Divergent: The History (and Future) of the International Joint Workshops in Computational Creativity. *AI Magazine* 30(3):15–22.
- Colton, S.; Llano, M. T.; Hepworth, R.; Charnley, J.; Gale, C. V.; Baron, A.; Pachet, F.; Roy, P.; Gervas, P.; Collins, N.; Sturm, B.; Weyde, T.; Wolff, D.; and Lloyd, J. 2016. The beyond the fence musical and computer says show documentary. In *7th International Conference on Computational Creativity (ICCC 2016)*. Paris (France): ACC.
- Colton, S. 2008. Creativity versus the Perception of Creativity in Computational Systems. In *Proceedings of AAAI Symposium on Creative Systems*, 14–20. Stanford, CA: AAAI.
- Csikszentmihalyi, M. 1988. Society, culture, and person: a systems view of creativity. In Sternberg, R. J., ed., *The Nature of Creativity*. Cambridge, UK: Cambridge University Press. chapter 13, 325–339.
- Davis, N.; Hsiao, C.-P.; Popova, Y.; and Magerko, B. 2015. An enactive model of creativity for computational collaboration and co-creation. In Zagalo, N., and Branco, P., eds., *Creativity in the Digital Age*. London, UK: Springer. 109–133.
- Davis, N. 2013. Human computer co-creativity: Blending human and computational creativity. In Smith, G., and Smith, A., eds., *Proceedings of the Doctoral Consortium of Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE) 2013*, number WS-13-23 in AAAI Technical Report, 9–12. Boston, MA: AAAI.
- Eigenfeldt, A.; Burnett, A.; and Pasquier, P. 2012. Evaluating musical metacreation in a live performance context. In *International Conference on Computational Creativity*, 140.
- Grace, K., and Maher, M. L. 2014. Towards computational co-creation in modding communities. In *Proceedings of AIIDE*. AAAI.
- Hennessey, B. A., and Amabile, T. M. 2010. Creativity. *Annual Review of Psychology* 61:569–598.
- Jacob, M., and Magerko, B. 2015. Interaction-based authoring for scalable co-creative agents. In Toivonen, H.; Colton, S.; Cook, M.; and Ventura, D., eds., *Proceedings of the Sixth International Conference on Computational Creativity*, 236–243. Park City, UT: ACC.
- Jordanous, A. 2012a. A Standardised Procedure for Evaluating Creative Systems: Computational Creativity Evaluation Based on What it is to be Creative. *Cognitive Computation* 4(3):246–279.
- Jordanous, A. 2012b. *Evaluating Computational Creativity: A Standardised Procedure for Evaluating Creative Systems and its Application*. Ph.D. Dissertation, University of Sussex, Brighton, UK.
- Jordanous, A. 2016. Four PPPPerspectives on computational creativity in theory and in practice. *Connection Science* 28(2):194–216.
- Jordanous, A. in press. Has computational creativity successfully made it ‘Beyond the Fence’ in musical theatre? *Connection Science*.
- Kantosallo, A.; Toivanen, J. M.; and Toivonen, H. 2015. Interaction evaluation for human-computer co-creativity: A case study. In *Proceedings of the 6th International Conference on Computational Creativity*. Park City, UT: ACC.
- Keller, R. M.; Toman-Yih, A.; Schofield, A.; and Merrit, Z. 2012. A creative improvisational companion based on idiomatic harmonic bricks. In Maher, M. L.; Hammond, K.; Pease, A.; Pérez y Pérez, R.; Ventura, D.; and Wiggins, G., eds., *Proceedings of the Third International Conference on Computational Creativity*, 155–159. ACC.
- Keller, R. M. 2012. Continuous improvisation and trading with Impro-Visor. In Maher, M. L.; Hammond, K.; Pease, A.; Pérez y Pérez, R.; Ventura, D.; and Wiggins, G., eds., *Proceedings of the Third International Conference on Computational Creativity*. ACC.
- Lamb, C.; Brown, D. G.; and Clarke, C. 2015. Human competence in creativity evaluation. In Toivonen, H.; Colton, S.; Cook, M.; and Ventura, D., eds., *Proceedings of the Sixth International Conference on Computational Creativity (ICCC 2015)*, 102–109. Park City, Utah: Brigham Young University.
- Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2012. Co-creating game content using an adaptive model of user taste. In Maher, M. L.; Hammond, K.; Pease, A.; Pérez y Pérez, R.; Ventura, D.; and Wiggins, G., eds., *Proceedings of the Third International Conference on Computational Creativity*. ACC.
- Lovelace, A. 1843. Notes on Manabrea’s Sketch of the Analytical Engine Invented by Charles Babbage. In Bowden, B., ed., *Faster than thought : a symposium on digital computing machines (1953)*. London: Pitman.
- Lubart, T. 2005. How can computers be partners in the creative process: Classification and commentary on the special issue. *International Journal of Human-Computer Studies* 63:365–369.
- Magerko, B.; Permar, J.; Jacob, M.; Comerford, M.; and Smith, J. 2014. An overview of computational co-creative pretend play with a human. In *Proceedings of the Playful Characters workshop at the Fourteenth Annual Conference on Intelligent Virtual Agents*.
- Maher, M. L. 2012. Computational and Collective Creativity: Who’s Being Creative? In *Proceedings of the 3rd International Conference on Computer Creativity*. Dublin, Ireland: ACC.
- Mamykina, L.; Candy, L.; and Edmonds, E. 2002. Collaborative creativity. *Communications of the ACM* 45(10):96–99.
- Meehan, J. R. 1976. *The metanovel: writing stories by computer*. Ph.D. Dissertation, Yale University, New Haven, CT, USA.
- Minsky, M. L. 1982. Why people think computers can’t. *AI Magazine* 3(4):3.
- Moffat, D. C. D. C., and Kelly, M. 2006. An investigation into people’s bias against computational creativity in music composition. In *The Third Joint Workshop on Computational Creativity*.
- Nake, F. 2009. Creativity in algorithmic art. In *Proceedings of the 2009 Conference on Creativity and Cognition*, 97–106.
- Pease, A., and Colton, S. 2011. On impact and evaluation in Computational Creativity: A discussion of the Turing Test and an alternative proposal. In *Proceedings of the AISB’11 Convention*. York, UK: AISB.
- Reffin-Smith, B. 2010. 43 Dodgy Statements on Computer Art. <http://zombiepataphysics.blogspot.com/2010/03/43-dodgy-statements-on-computer-art.html>, last accessed 29th July 2010.
- Rhodes, M. 1961. An analysis of creativity. *Phi Delta Kappan* 42(7):305–310.
- Schneiderman, B. 2007. Creativity support tools: accelerating discovery and innovation. *Communications of the ACM* 50(12):20–32.
- Yannakakis, G. N.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-initiative co-creativity. In *Proceedings of the 9th Conference on the Foundations of Digital Games*.

DeepTingle

Ahmed Khalifa, Gabriella A. B. Barros and Julian Togelius

Department of Computer Science and Engineering

New York University

Brooklyn, NY 11201 USA

ahmed.khalifa@nyu.edu, gabriella.barros@gmail.com and julian@togelius.com

Abstract

DeepTingle is a text prediction and classification system trained on the collected works of the renowned fantastic gay erotica author Chuck Tingle. Whereas the writing assistance tools you use everyday (in the form of predictive text, translation, grammar checking and so on) are trained on generic, purportedly “neutral” datasets, DeepTingle is trained on a very specific, internally consistent but externally arguably eccentric dataset. This allows us to foreground and confront the norms embedded in data-driven creativity and productivity assistance tools. As such tools effectively function as extensions of our cognition into technology, it is important to identify the norms they embed within themselves and, by extension, us. DeepTingle is realized as a web application based on LSTM networks and the GloVe word embedding, implemented in JavaScript with Keras-JS.

Introduction

We live continuously computationally assisted lives. Computational assistance tools extend and scaffold our cognition through the computational devices, such as phones and laptops, that many of us keep close at all times. A trivial-seeming but important example is predictive text entry, also popularly known as autocomplete. The absence of regular keyboards on mobile devices have necessitated software which maps button-presses (or swipes) to correct words, and thus guesses what word we meant to write. In many cases, e.g. on the iPhone, the software also guesses what word you plan to write next and gives you the chance to accept the software’s suggestion instead of typing the word yourself. Even when writing on a computer with a real keyboard, spell-checking software is typically running in the background to check and correct the spelling and sometimes the grammar of the text. In the structured domain of programming, Integrated Development Environments such as Eclipse or Visual Studio suggest what methods you want to call based on data-driven educated guesses. Relatedly, when shopping or consuming music or videos online, recommender systems are there to provide us with ideas for what to buy, watch or listen to next.

Beyond the relatively mundane tasks discussed above, there is a research vision of computational assistance with more creative tasks. The promise of computational creati-

ty assistance tools is to help human beings, both professional designers and more casual users, to exercise their creativity better. An effective creativity assistance tool helps its users be creative by, for example, providing domain knowledge, assisting with computational tasks such as pattern matching, providing suggestions, or helping enforce constraints; and many other creativity assistance mechanisms are possible. This vision is highly appealing for those who want to see computing in the service of humanity. In the academic research community, creativity assistance tools are explored for such diverse domains as music (Hoover, Szerlip, and Stanley 2011), game levels (Liapis, Yannakakis, and Togelius 2013; Smith, Whitehead, and Mateas 2011; Shaker, Shaker, and Togelius 2013), stories (Roemmele and Gordon 2015), drawings (Zhang et al. 2015), and even ideas (Llano et al. 2014).

There’s no denying that many of these systems can provide real benefits to us, such as faster text entry, useful suggestion for new music to listen to, or the correct spelling for Massachusetts. However, they can also constrain us. Many of us have experienced trying to write an uncommon word, a neologism, or a profanity on a mobile device just to have it “corrected” to a more common or acceptable word. Word’s grammar-checker will underline in aggressive red grammatical constructions that are used by Nobel prize-winning authors and are completely readable if you actually read the text instead of just scanning it. These algorithms are all too happy to shave off any text that offers the reader resistance and unpredictability. And the suggestions for new books to buy you get from Amazon are rarely the truly left-field ones—the basic principle of a recommender system is to recommend things that many others also liked.

What we experience is an algorithmic enforcement of norms. These norms are derived from the (usually massive) datasets the algorithms are trained on. In order to ensure that the data sets do not encode biases, “neutral” datasets are used, such as dictionaries and Wikipedia. (Some creativity support tools, such as Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2013), are not explicitly based on training on massive datasets, but the constraints and evaluation functions they encode are chosen so as to agree with “standard” content artifacts.) However, all datasets and models embody biases and norms. In the case of everyday predictive text systems, recommender systems and so on, the

model embodies the biases and norms of the majority.

It is not always easy to see biases and norms when they are taken for granted and pervade your reality. Fortunately, for many of the computational assistance tools based on massive datasets there is a way to drastically highlight or foreground the biases in the dataset, namely to train the models on a completely different dataset. In this paper we explore the role of biases inherent in training data in predictive text algorithms through creating a system trained not on “neutral” text but on the works of Chuck Tingle.

Chuck Tingle is a renowned Hugo award nominated author of fantastic gay erotica. His work can be seen as erotica, science fiction, absurdist comedy, political satire, met-aliterature, or preferably all these things and more at the same time. The books frequently feature gay sex with unicorns, dinosaurs, winged derrires, chocolate milk cowboys, and abstract entities such as Monday or the very story you are reading right now. The bizarre plotlines feature various landscapes, from paradise islands and secretive science labs, to underground clubs and luxury condos inside the protagonist’s own posterior. The corpus of Chuck Tingle’s collected works is a good choice to train our models on precisely because they so egregiously violate neutral text conventions, not only in terms of topics, but also narrative structure, word choice and good taste. They are also surprisingly consistent in style, despite the highly varied subjects. Finally, Chuck Tingle is a very prolific author, providing us with a large corpus to train our models on. In fact, the consistency and idiosyncrasy of his literary style together with his marvelous productivity has led more than one observer to speculate about whether Chuck Tingle is actually a computer program, an irony not lost on us.

In this paper, we ask the question what would happen if our writing support systems did not assume that we wanted to write like normal people, but instead assumed that we wanted to write like Chuck Tingle. We train a deep neural net based on Long Short-Term Memory and word-level embeddings to predict Chuck Tingle’s writings, and using this model we build a couple of tools (a predictive text system and a reimagining of literary classics) that assists you with getting your text exactly right, i.e. to write just like Chuck Tingle would have.

A secondary goal of the research is to investigate how well we can learn to generate text that mimics the style of Chuck Tingle from his collected works. The more general question is that of generative modeling of literary style using modern machine learning methods. The highly distinctive style of Tingle’s writing presumably makes it easy to verify whether the generated text adheres to his style.

Background

This work builds on a set of methods from modern machine learning, in particular in the form of deep learning.

Word Embedding

Word embedding is a technique for converting words into a n-dimensional vector of real numbers, capable of capturing probabilistic features of the words in the current text.

The primary goal is to reduce the dimensionality of the word space to a point where it can be easily processed. Each dimension in the vector represent a linguistic context, and the representation should preserve characteristics of the original word (Goldberg and Levy 2014).

Such mappings have been achieved using various techniques, such as neural networks (Bengio, Ducharme, and Vincent 2003), principal component analysis (Lebret and Collobert 2013), and probabilistic models (Globerson et al. 2007). A popular method is skip-gram with negative-sampling training, a context-predictive approach implemented in word2vec models (Mikolov et al. 2013). On the other hand, global vectors (GloVe) is a context-count word embedding technique (Pennington, Socher, and Manning 2014). GloVe captures the probability of a word appearing in a certain context in relation to the remaining text.

Neural Networks and Recurrent Neural Networks

Neural networks (NN) are a machine learning technique originally inspired by the way the human brain functions (Hornik, Stinchcombe, and White 1989). The basic unit of a NN is a neuron. Neurons receive vectors as inputs, and output values by applying a non linear function to the multiplication of said vectors and a set of weights. They are usually grouped in layers, and neurons in the same layer cannot be connected to each other. Neurons in a given layer are fully connected to all neurons in the following layer. NNs can be trained using the backpropagation algorithm. Backpropagation updates the network weights by taking small steps in the direction of minimizing the error measured by the network.

A recurrent neural network (RNN) is a special case of neural network. In a RNN, the output of each layer depends not only on the input to the layer, but also on the previous output. RNNs are trained using backpropagation through time (BPTT) (Werbos 1990), an algorithm that unfolds the recursive nature of the network for a given amount of steps, and applies a generic backpropagation to the unfolded RNN. Unfortunately, BPTT doesn’t suit vanilla RNNs when they run for large amount of steps (Hochreiter 1998). One solution for this problem is the use of Long Short-Term Memory (LSTM). LSTMs were introduced by Sepp Hochreiter and Jürgen Schmidhuber (1997), and introduces a memory unit. The memory unit acts as a storage device for the previous input values. The input is added to the old memory state using gates. These gates control the percentage of new values contributing to the memory unit with respect to the old stored values. Using gates helps to sustain constant optimization through each time step.

Natural Language Generation

Natural language generation approaches can be divided into two categories: Rule- or template-based and machine learning (Tang et al. 2016). Rule-based (or template-based) approaches (Cheyer and Guzzoni 2014; Mirkovic and Cavedon 2011) were considered norm for most systems, with rules/templates handmade. However, these tend to be too specialized, not generalizing well to different domains, and a large amount of templates is necessary to gen-

erate quality text even on a small domain. Some effort has been made towards generating the template based on a corpus, using statistical methods (Mairesse et al. 2010; Mairesse and Young 2014; Oh and Rudnicky 2000), but these still require a large amount of time and expertise.

Machine learning, in particular RNNs, has become an increasingly popular tool for text generation. Sequence generation by character prediction has been proposed using LSTM (Graves 2013)) and multiplicative RNNs (Sutskever, Martens, and Hinton 2011). Tang et al. (2016) attempted associating RNNs and context-awareness in order to improve consistency, by encoding not only the text, but also the context in semantic representations. Context has also been applied in response generation in conversation systems (Sordani et al. 2015; Wen et al. 2015b).

Similarly, machine learning is also used in machine translation (Sutskever, Vinyals, and Le 2014; Cho et al. 2014; Bahdanau, Cho, and Bengio 2014). These approaches tend to involve training a deep network, capable of encoding sequences of text from an original language in a fixed-length vector, and decoding output sequences to the targeted language.

Creativity Assistance Tools

Several works have been proposed to foster the collaboration between machine and user in creative tasks. Goel and Joyner argue that scientific discovery can be considered a creative task, and propose MILA-S, an interactive system with the goal of encouraging scientific modeling (Goel and Joyner 2015). It makes possible the creation of conceptual models of ecosystems, which are evaluated with simulations.

CAHOOTS is a chat system capable of suggesting images as possible jokes (Wen et al. 2015a). STANDUP (Waller et al. 2009) assists children who use augmentative and alternative communication to generate puns and jokes.

Co-creativity systems can also help the creation of fictional ideas. Llano et al. (2014) describe three baseline ideation methods using ConceptNet, ReVerb and bisociative discovery, while I-get (Ojha, Lee, and Lee 2015) uses conceptual and perceptual similarity to suggest pairs of images, in order to stimulate the generation of ideas.

DrawCompileEvolve (Zhang et al. 2015) is a mixed-initiative art tool, where the user can draw and group simple shapes, and make artistic choices such as symmetric versus asymmetric. The system then uses uses neuroevolution to evolve a genetic representation of the drawing.

Sentient Sketchbook and Tanagra assist in the creation of game levels. Sentient Sketchbook uses user-made map sketches to generate levels, automate playability evaluations and provide various visualizations (Liapis, Yannakakis, and Togelius 2013; Yannakakis, Liapis, and Alexopoulos 2014). Tanagra uses the concept of rhythm to generate levels for a 2D platform (Smith, Whitehead, and Mateas 2010).

Focusing on writing, we can highlight the Poetry Machine (Kantosalo et al. 2014) and Creative Help (Roemmele and Gordon 2015). Both aim to provide suggestions to writers, assisting their writing process. The Poetry Machine creates draft poems based on a theme selected by the user. Creative

Help uses case-based reasoning to search a large story corpus for possible suggestions (Roemmele and Gordon 2015).

DeepTingle

This section discusses the methodology applied in DeepTingle. DeepTingle consists of two main components: the neural network responsible for the learning and prediction of words in the corpus, and a set of co-creativity tools aimed at assisting in the writing or style-transfer of text. The tools described (Predictive Tingle and Tingle Classics) are available online, at <http://www.deeptingle.net>.

Our training set includes all Chuck Tingle books released until November 2016: a total of 109 short stories and 2 novels (with 11 chapters each) to create a corpus of 3,044,178 characters. The text was preprocessed by eliminating all punctuation, except periods, commas, semicolons, question marks and apostrophes. The remaining punctuation marks, excluding apostrophes, were treated as separate words. Apostrophes were attached to the words they surround. For example, "I'm" is considered a single word.

Network Architecture

We experimented with different architectures. Our initial intuition was to mimic the architecture of different Twitter bots. Twitter's limitation of 140 characters per tweet influenced the strategy used by most neural network trained bots. They tend to work on a character-by-character approach, producing the next character based on previous characters, not words. Similarly, our first architecture, shown in Figure 1, was inspired by this representation. The numbers in the figure represent the size of data flows between network layers. The neural network consists of 3 layers: 2 LSTM layers followed by a softmax one. A softmax layer uses softmax function to convert the neural network's output to the probability distribution of every different output class (Bridle 1990). In our case, classes are different letters. The size of input and output is 57, because that's the total number of different characters in Chuck Tingle's novels. Input is represented as one hot encoding, which represents data as a vector of size n , where $n - 1$ values are 0's, and only one value is 1, signaling the class the input belongs to.

After initial testing, we opted to switch to a word representation instead of character representation. While word-based architectures repress the network's ability of creating new words, they leverage the network's sequence learning. Figure 2 shows the current architecture used in DeepTingle.

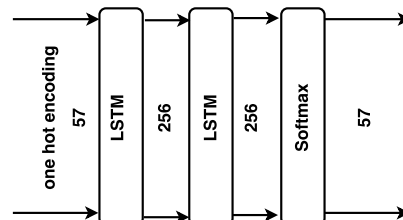


Figure 1: Alphabet based neural network architecture used in DeepTingle.

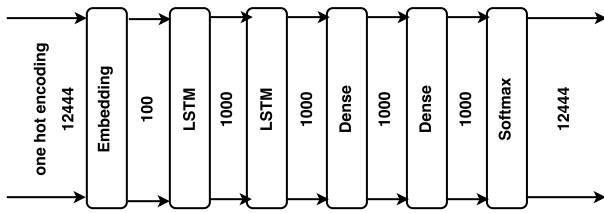


Figure 2: Word-based neural network architecture used in DeepTingle.

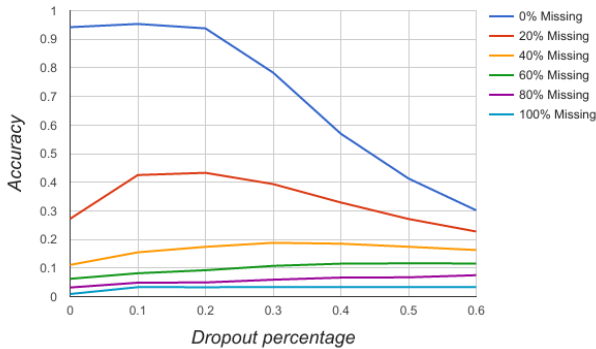


Figure 3: Graph shows the effect of using dropout against noise.

The network consists of 6 layers. The first layer is an embedding one that converts an input word into its 100 dimension representation. It is followed by 2 LSTM layers of size 1000, which in turn are followed by 2 fully connected layers of same size. Finally, there is a softmax layer of size 12,444 (the total number of different words in all Tingle’s books).

Network training

The network training consisted of two phases. The first one aims at training the embedding layer separately, using GloVe and all Chuck Tingle’s stories in the corpus. In the second phase, we trained the remaining part of the network. Our reasoning for such approach was to speed up the learning process. Dropout is used as it increase the network accuracy against unknown input words (missing words). Figure 3 shows the effect of the dropout on the network accuracy. The graph shows using 20% as a dropout value gives the highest accuracy without sacrificing any accuracy at 0% missing words.

We use a recently proposed optimization technique, the Adam Optimizer (Kingma and Ba 2014), to train the network, with a fixed learning rate (0.0001). This technique reaches a minimum value faster than traditional backpropagation. We experimented with various amount of time steps for the LSTM and settled for 6 time steps, for it generated sentences that were more grammatically correct and more coherent than the other experiments. Input data is designed to predict the next word based on the previous 6 words.

Predictive Tingle

Predictive Tingle is a writing support tool built on top of the previously mentioned network. Its goal is to provide suggestions of what next word to write, based on what the user has written so far. It does so by preprocessing and encoding the user’s input, feeding it to the network, and decoding the highest ranked outputs, which are shown as suggestions.

As the user writes, the system undergoes two phases: substitution and suggestion. Whenever a new word is written, Predictive Tingle verifies if the word appears in a Tingle-nary, a dictionary of all words from Chuck Tingle’s books. If the word appears, nothing changes in this step. Otherwise, the system searches for the word in the dictionary closest to the input, using Levenshtein’s string comparison (Levenshtein 1966). The input is then replaced with said word.

Once the substitution phase ends, the system searches for possible suggestions. It uses the last 6 written words as input for the trained network, and suggest the word with the highest output. The user can then accept or reject the suggestion. If he/she accepts, either by pressing the ‘Enter’ key of clicking on the suggestion button, the word is inserted in the text, and the system returns to the beginning of the suggestion phase. Otherwise, once a new word is written, the system returns to the substitution phase.

Tingle Classics

Tingle Classics aims to answer the question: “what would happen if classic literature was actually written by Chuck Tingle?” The user can select one line from a series of opening lines from famous and/or classic books (e.g. *1984* by George Orwell, or *Moby-dick* by Herman Melville). The system uses the line to generate a story, by repeatedly predicting the next word in a sentence. The user can also parameterize the amount of words generated, and whether to transform words that aren’t in Tingle’s works into words from the corpus.

Results

This section presents our results regarding the neural network training, an user study, and the two co-creativity tools developed (Predictive Tingle and Tingle Classics). A third tool, called Tingle Translator, aimed at transferring Chuck Tingle’s style of writing to any given text using NN and word embeddings. Unfortunately, the embedding space for Chuck Tingle’s novels is too small in comparison to the word embedding trained from Wikipedia articles. This led to a failed attempt to have a meaningful relation between both embeddings. Using a neural network to bridge this gap wasn’t a success, and as such Tingle Translator will not be discussed further in this work, remaining a possibility for future work.

Network Training

DeepTingle trained for 2,500 epochs using the Adam Optimizer with fixed learning rate 0.0001. After 2000 epochs there was no improvement in loss. The network reached accuracy of 95% and an error drop from 12.0 to 0.932.

We experimented with different sizes of word sequences, from 1 word up to 20 words. Examples 1 and 2 show chunks

Example 1 Generated story where every new word depends on the previous 6 words.

I was walking in the streets going to my friend's house. While I was walking, I stumbled upon the chamber and then heading out into the parking lot and calling my girlfriend to confirm my status as a normal, red blooded, American heterosexual. yet, despite my best efforts, I find myself getting turned on. whoa. Kirk says with a laugh, sensing the hardening of my cock up against his back. You getting excited back there, buddy? No. I protest, defensively. It sure doesn't feel like it. The unicorn prods with a laugh. That feels like a big fucking human cock pressed up against my back. I don't say a word, completely embarrassed. You ever fucked a unicorn? Kirk asks me suddenly. I can immediately sense a change in his tone, a new direction in his unicorn mannerisms all the way down to the way the he turns his large beastly head to speak to me. No, I can't say that i have. I explain. You're the first one I've met. Kirk nods. Yep, there's not a lot of us out there, not a lot of gay one's either.

Example 2 Generated story where every new word depends on the previous 20 words.

I was walking in the streets going to my friend's house. While I was walking , I stumbled upon the hustle and bustle of my surroundings. instead of my win, i begin to weave out into the air with a second moments, eventually my discomfort becomes apparent and closer to the cars. suddenly, i feel the strangely gay being of chibs suddenly, only this long i try not to stare too. where am i like? i question. but, you have a point, jonah says. when i was in there for a moment, my mind drifting almost i have ever seen in this situation; no living longer in our game. as i said this was the hunk hand, and i know this about the man in a situation so much more than i have to really right about this. i understand, that's how i want to do and handsome, love. of course, it is, i really believe that i really want. ever before, i don't know. my wife explains, the rich man explains. this was amazing, i remind him. the dinosaur takes a few steps behind the top of the stage and immediately standing up the front screen.

of generated text in 2 sizes (6 and 20 word sequence). All experiments started with the same input, i.e. "I was walking in the streets going to my friend's house . While I was walking , I stumbled upon", and generated at least 200 words. It is trivial to recognize that the 6 words sequence produce more grammatically correct sentences compared to the 20 words sequence. On the other hand, 20 words sequences have higher chance to refer to something that happened before, and less chances of getting stuck in loops when compared to 6 words sequences.

To better understand the effect of increasing the sequence size, we generated a 200,000 words text, to be compared to original Chuck Tingle stories in order to evaluate how similar they are. The similarity is calculated by counting the number of identical sequence of words between the generated text and the original text. Figure 4 shows the different N-Grams for all the sequence sizes. The 4-words sequence

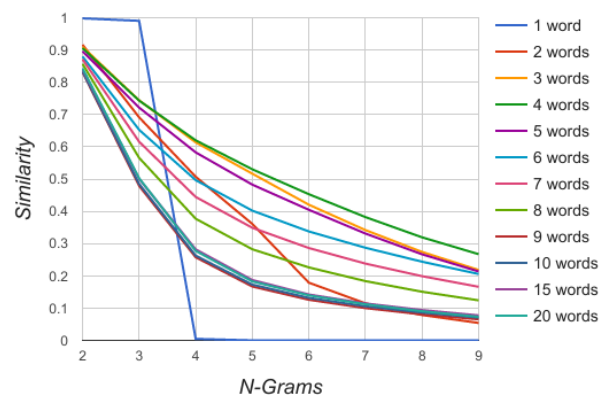


Figure 4: Graph with the similarity between generated texts and the actual chuck tingle stories for all 4 sequence sizes.

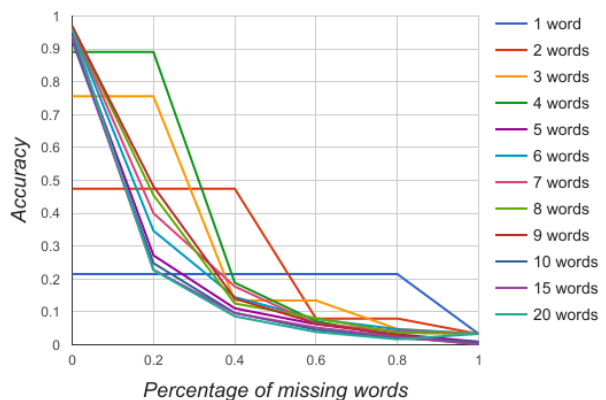


Figure 5: This graph is showing the robustness of the network against missing information for all 4 sequence sizes.

is the most similar to original Chuck Tingle text. Interestingly, all sizes above 8 words have the same amount of similarity. We believe this may be due to the LSTM reaching its maximum capacity at size of 9.

Another experiment aimed at testing the robustness of the network, by testing the effect of unknown words on the accuracy of prediction. Figure 5 describes the accuracy for all the sequence sizes against different percentages of missing words from the input text. It shows that the more words we have the better the results except for sizes 3 and 4. At these sizes, 20% missing data means nothing change. We chose size 6 as it is higher than the others, and at the same time won't compromise the neural network speed.

User Study

We performed a user study to compare the generated text by DeepTingle to Chuck Tingle's original text. Additionally, we wanted to confirm if a neural network would actually have an advantage over a simpler representation, such as a Markov chain model. We trained a Markov chain on the

	Grammar	Coherence	Interesting
CT vs DT	16/23*	19/27*	17/31
CT vs Markov	29/31**	31/33**	26/33**
DT vs Markov	17/21**	21/27**	11/19

Table 1: Table shows the result of the user study where **CT** is Chuck Tingle’s original text, **Markov** is the Markov chain generated text, and **DT** is the DeepTingle generated text. The superscript indicate the p-value from using binomial test. * indicated that the p-value is less than 5%, while ** indicates the p-value is less than 1%.

same data set, and chose the state size to be 3 as it empirically achieved the best results without losing generalization ability.

In the user study, the user is presented with two pieces of text of equal length picked randomly from any of the 3 categories of text (Chuck Tingle’s original text, DeepTingle text, and Markov chain text). The user has to answer 3 questions: “Which text is more grammatically correct?”; “Which text is more interesting?”; and “Which text is more coherent?”. The user could pick one of four options: “Left text is better”, “Right text is better”, “Both are the same”, or “None”.

We collected approximately 146 different comparisons. Table 1 presents the results of comparisons, excluding all choices for “Both are the same” or “None of them”. The values represent the fraction of times the first text is voted over the second one. Results show that using neural networks for text prediction produce more coherent and grammatically correct text than Markov chain, but less so than the original text, which is reasonable considering the latter is written and reviewed by a human.

Predictive Tingle

Figure 6 shows a screenshot of the system: On top we have a brief description of what Predictive Tingle is. Right below, a text field where the user can write text. To the text field’s right, a purple suggestion button that is updated every time the user presses the spacebar. In this example, the user wrote “*It was raining in New York*”, and pressed enter consecutively, allowing the system to finish the input. The outcome was “*It was raining in New York city. It’s not long before the familiar orgasmic sensations begin to bubble up within me once again, spilling out through my veins like simmering erotic venom.*”

Tingle Classics

The final part of the tools is Tingle Classics, shown in Figure 7. From top to bottom, the screen shows the tool’s name and description, followed by a list of books, to be selected by the user. A button, “Generate!”, triggers the word generation. A line, right below the bottom, shows the original initial line for the book selected. Two configurations options can be found in sequence: the option of toggle substitution on and off, and the amount of words to generate. Finally, the story generated is outputted at the very bottom of the page.

If substitution is selected, a preprocessing of the initial line is made, transforming every word in the original text

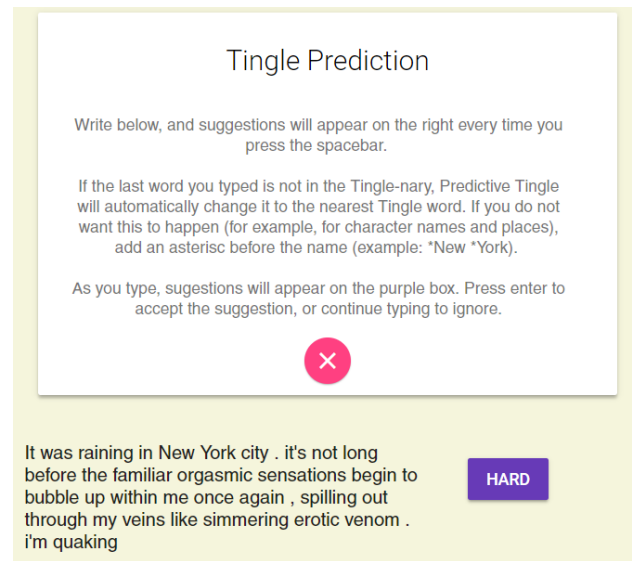


Figure 6: Screenshot of Predictive Tingle. Shows the input box with an example text, and a suggestion of the next word.

that doesn’t appear in the Tingle corpus, into a Tingle word. Thus, it guarantees that every word in the input vector appears in the Tingle corpus. If substitution is not used, words not in the Tingle corpus are skipped. For example, if the sentence is “Hello, my name is Inigo Montoya”, and neither “Inigo” nor “Montoya” belong in the corpus, the vector would shift to embed only “Hello, my name is” (notice that the comma is considered a word). This may result in diverging stories, as shown in Examples 3 and 4. Both are generated from the same line (“Call me Ishmael”, from Moby-Dick, by Herman Melville), but the first doesn’t use substitution, while the second does.

Example 3 150 words generated from the line “Call me Ishmael”, without word substitution.

Call me ishmael a simple season. The creature declares, driving the rest of his drink and then gets it in, his eyes watering tight as he thrusts into me, the massive rod filling my butthole entirely as i cry out with a yelp of pleasure. Colonel peach wastes no time now, immediately getting to work as he rams my body from behind. I grip tightly onto the bed sheets in front of me, bracing myself against the hood as slater continues to pump in and out of my butt, slowly but firmly as i tremble from his skilled touch. My legs are spread wide for him, held back as he slams into me at an ever escalating speed. Soon enough, kirk is hammering into me with everything he’s got, his hips pounding loudly against the side of the boulder

Conclusion and Future Work

This paper proposes a two-part system, composed of a deep neural network trained over a specific literary corpus and a writing assistance tool built on the network. Our corpus

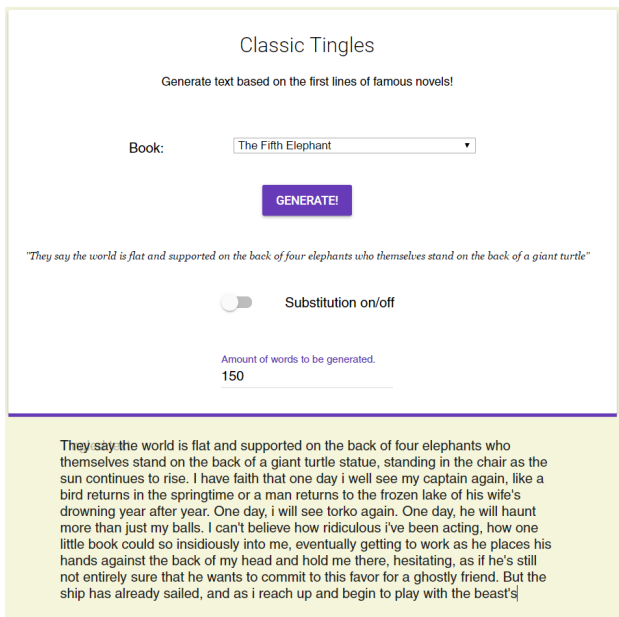


Figure 7: Display of Tingle Classics, generating 150 words from the first lines in Terry Pratchett’s “The Fifth Elephant”.

consists solely of works by renowned author Chuck Tingle. This corpus represents a large set of stories, diverse in setting and context, but similar in structure. Its controversial themes negates the “neutral” norm of writing assistance tools currently available. We trained a six layer architecture, using GloVe embedding, LSTMs, dense and softmax layers, capable of word sequence prediction. Our system allows for users to write stories, receiving word suggestions in real time, and to explore the intersection of classic literature and the fantastic erotic niche that Tingle embodies.

We are excited to study how much deeper we can take DeepTingle. We intend to improve the system’s architecture, in order to increase its prediction accuracy against missing words. Furthermore, a possibility is to incorporate generative techniques to evolve grammars based on Tingle’s work. Additionally, we intend on improving and adding new co-creativity tools, in particular the Tingle Translator. The use case of the Tingle Translator is to take existing English text and translate it to Tingle’s universe by substituting commonly used but un-Tingly words and phrases with their Tingle-equivalents. For this, we will explore different approaches to map words into embedding space, including the use of bidirectional networks and style transfer.

The central idea motivating this study and paper was to expose the norms inherent in “neutral” corpuses used to train AI-based assistants, such as writing assistants, and explore what happens when building a writing assistance tool trained on very non-neutral text. It is very hard to gauge the success of our undertaking through quantitative measures such as user studies. We believe that the effects of DeepTingle can best be understood by interacting with it directly, and we urge our readers to do so at their leisure.

Example 4 150 words generated from the line “Call me Ishmael”, using word substitution.

Call me small new era of the night before, but somehow my vision is assaulted by sudden and graphic depictions of gay sex. I scramble to change the channel and quickly realize that every station has been somehow converted into hardcore pornography. What the fuck? I ask in startled gasp. What is this? I know that we both have a knack for running out on relationships. Portork tells me. But we also know love when we see it. A broad smile crosses my face. I see you’ll also picked up my habit of inappropriate practical jokes. Portork laughs. Of course. Now get in here an fuck me, it’s time for round two. Oliver explains. And i may be a country boy but i’m not stupid. I might not have the password or whatever it is that

Acknowledgments

We thank Marco Scirea, for helping us conceive ideas for this work, Philip Bontrager, for useful discussions, Scott Lee and Daniel Gopstein, for their support and enthusiasm. We gratefully acknowledge a gift of the NVidia Corporation of GPUS to the NYU Game Innovation Lab. Gabriella Barros acknowledges financial support from CAPES and the Science Without Borders program, BEX 1372713-3. Most of this paper was written by humans.

References

- Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Bengio, Y.; Ducharme, R.; and Vincent, P. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.
- Bridle, J. S. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*. Springer. 227–236.
- Cheyner, A., and Guzzoni, D. 2014. Method and apparatus for building an intelligent automated assistant. US Patent 8,677,377.
- Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.
- Globerson, A.; Chechik, G.; Pereira, F.; and Tishby, N. 2007. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research* 8(Oct):2265–2295.
- Goel, A. K., and Joyner, D. A. 2015. Impact of a creativity support tool on student learning about scientific discovery processes. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 284.
- Goldberg, Y., and Levy, O. 2014. word2vec explained: Deriving mikolov et al.’s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.
- Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

- Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.
- Hochreiter, S. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(02):107–116.
- Hoover, A. K.; Szerlip, P. A.; and Stanley, K. O. 2011. Interactively evolving harmonies through functional scaffolding. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 387–394. ACM.
- Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5):359–366.
- Kantosalo, A.; Toivanen, J. M.; Xiao, P.; and Toivonen, H. 2014. From isolation to involvement: Adapting machine creativity software to support human-computer co-creation. In *Proceedings of the Fifth International Conference on Computational Creativity*, 1–8.
- Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Lebret, R., and Collobert, R. 2013. Word embeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*.
- Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, 707–710.
- Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *FDG*, 213–220.
- Llano, M. T.; Hepworth, R.; Colton, S.; Gow, J.; Charnley, J.; Lavrac, N.; Znidaršič, M.; Perovšek, M.; Granroth-Wilding, M.; and Clark, S. 2014. Baseline methods for automated fictional ideation. In *Proceedings of the 5th international conference on computational creativity*.
- Mairesse, F., and Young, S. 2014. Stochastic language generation in dialogue using factored language models. *Computational Linguistics*.
- Mairesse, F.; Gašić, M.; Jurčiček, F.; Keizer, S.; Thomson, B.; Yu, K.; and Young, S. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1552–1561. Association for Computational Linguistics.
- Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.
- Mirkovic, D., and Cavedon, L. 2011. Dialogue management using scripts. US Patent 8,041,570.
- Oh, A. H., and Rudnicky, A. I. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, 27–32. Association for Computational Linguistics.
- Ojha, A.; Lee, H.-K.; and Lee, M. 2015. I-get: A creativity assistance tool to generate perceptual pictorial metaphors. In *Proceedings of the 3rd International Conference on Human-Agent Interaction*, 311–314. ACM.
- Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–1543.
- Roemmele, M., and Gordon, A. S. 2015. Creative help: a story writing assistant. In *International Conference on Interactive Digital Storytelling*, 81–92. Springer.
- Shaker, N.; Shaker, M.; and Togelius, J. 2013. Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels. In *AIIDE*.
- Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 209–216. ACM.
- Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):201–215.
- Sordani, A.; Galley, M.; Auli, M.; Brockett, C.; Ji, Y.; Mitchell, M.; Nie, J.-Y.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.
- Sutskever, I.; Martens, J.; and Hinton, G. E. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1017–1024.
- Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.
- Tang, J.; Yang, Y.; Carton, S.; Zhang, M.; and Mei, Q. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900*.
- Waller, A.; Black, R.; OMara, D. A.; Pain, H.; Ritchie, G.; and Manurung, R. 2009. Evaluating the standup pun generating software with children with cerebral palsy. *ACM Transactions on Accessible Computing (TACCESS)* 1(3):16.
- Wen, M.; Baym, N.; Tamuz, O.; Teevan, J.; Dumais, S.; and Kalai, A. 2015a. Omg ur funny! computer-aided humor with an application to chat. In *Proceedings of the 6th International Conference on Computational Creativity, ICC3*, 86–93.
- Wen, T.-H.; Gasic, M.; Mrksic, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.
- Werbos, P. J. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560.
- Yannakakis, G. N.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-initiative co-creativity. In *FDG*.
- Zhang, J.; Taarnby, R.; Liapis, A.; and Risi, S. 2015. Draw-compileevolve: Sparking interactive evolutionary art with human creations. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, 261–273. Springer.

Fluidic Games in Cultural Contexts

Mark J. Nelson, Swen E. Gaudi, Simon Colton, Edward J. Powley,
Blanca Pérez Ferrer, Rob Saunders, Peter Ivey, Michael Cook

The MetaMakers Institute
Falmouth University
metamakersinstitute.com

Abstract

We introduce fluidic games, a type of casual creator that blends game play and game design. Fluidic games have a core of built-in games that anchor a space of design possibilities around them, and encourage players to alternate between playing specific games and playing with the design space. Our Gamika Technology platform supports fluidic games on mobile devices, and we have thus far built three of them. In doing so, we have found that even for simple games, fluidic games require computational creativity support. This takes several forms intended to keep design sessions playful and fast-moving, including automated game design used as a form of brainstorming, mixed-initiative co-creative design to ease design-space navigation, and automated game playing to evaluate game dynamics. Finally, we have exhibited this fluidic-games concept in three distinct cultural settings: a series of rapid game jams lasting 1–2 hours each, an in-progress semester-long enrichment course with a local school, and an art installation that foregrounds an autonomous version of the system exploring a fluidic game on its own, at least if the audience will allow it to do so.

Introduction

Fluidic games are initially just games, playable as any other game. But in contrast to games that emphasise a single, carefully designed artefact, fluidic games emphasise that any individual game is always only a single point in a larger game-design space, from which many other games could also have been made, from trivial variants to significantly different games. Players are encouraged to explore this space with minimal context shift between the playing and design-exploration modes.

A focus of our research agenda is investigating different approaches to this exploration, which can be viewed as falling along the spectrum of mixed-initiative human/machine co-creativity in games (Smith, Whitehead, and Mateas 2011; Grace and Maher 2014; Yannakakis, Liapis, and Alexopoulos 2014; Liapis, Smith, and Shaker 2016; Nelson et al. 2017). At one end of the spectrum of mixed-initiative creativity is an orientation towards enabling human creativity (Shneiderman 2007); at the other is fully autonomous game creation (Cook, Colton, and Gow 2016).

We focus on casual games played on mobile devices, which many people play, but few design. We aim to help to democratise this situation by making the player/designer boundaries more fluid, so players can play individual games and also play with the design itself, within the same app and with frequent alternation between the two modes. In addition to minimal context shift from playing to designing, we also aim to have a difference in time commitment: users playing an iPhone game on the bus ride home should be able to spend 20 minutes designing a variation of that game on their iPhone, too, and then go back to playing their new game. Or, they might want to press a button and have an AI designer generate a new game—we have found that even when oriented towards human design, some degree of automated design is desirable to make navigating the space playful rather than tedious. A fluidic game is not just a game to play, but neither is it a traditional game-design tool.

Fluidic games therefore fit into the larger category of accessible, low-commitment, fun-to-use creative tools dubbed casual creators (Compton and Mateas 2015). They also fit into a broader class of user-modifiable games, which we'll call *maker-games*, which give players the ability to change some aspect of the game, most commonly by including a level editor (as seen in Nintendo's *Super Mario Maker*).

Mobile games are an especially good setting for casual creators, both because they are widely played even by people who don't necessarily see themselves as "creators", and because games foreground concepts such as initiative and agency that allow creative game design tools to piggyback on familiar game-playing terminology and concepts. Games also pose a challenge by integrating many creative design domains, from systems thinking to storytelling to visual aesthetics (Liapis, Yannakakis, and Togelius 2014).

We have piloted the fluidic-games concept in three distinct cultural settings, in addition to planning the public release of two fluidic games, *Wevva* and *No Second Chance*, on the iOS App Store. We have used both *Wevva* and *No Second Chance* to host rapid game jams, a version of a game jam in which players can make their own games in as little as 10 minutes, with the overall jam lasting no more than 1–2 hours (traditional game jams typically last 24–72 hours). We are currently engaged in a more extended educational experiment using *No Second Chance* to teach game-design and elementary physics principles to students in a lo-

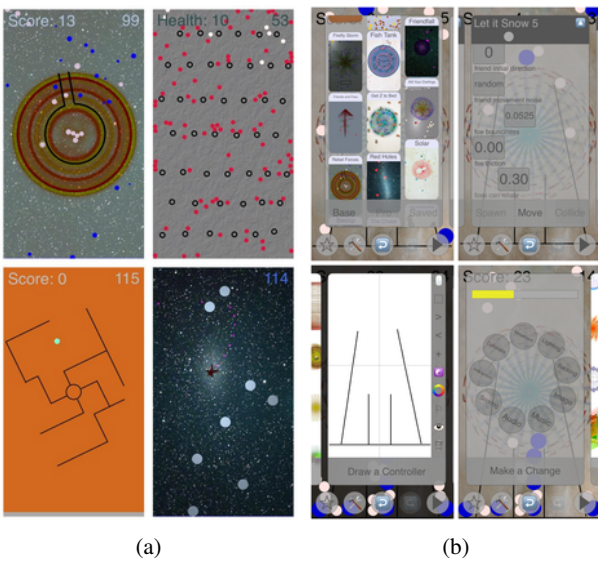


Figure 1: (a) Four Gamika games designed with (b) *Cillr*. *Cillr* design panels clockwise from top left: List of editable saved games, a screen of movement-related sliders, brainstorming wheel to randomise subsets of parameters, and drawing interface to edit controllers.

cal school. And finally, we designed and exhibited an artistic installation called *I Create, You Destroy*, based on a fully autonomous version of *No Second Chance* in which an autonomous creative system designs and plays its own games, and human participants (if they choose to participate) play only a destructive role in the creative process.

Gamika Technology

To enable the design of fluidic games, we have built a platform, Gamika Technology, that parameterises a game-design space with 284 parameters, plus associated visual and audio assets. Parametric design is not the only technique for building fluidic games, but one we think is well suited to the task, as it casts the problem of design-space navigation in a concrete setting suited to both user-interface design and automated exploration. On this platform, we have built three fluidic games. One, *Cillr*, is based on the entire space and is used primarily as an in-house app to explore the design space in full generality; while initially intended as a fluidic game, it may be seen as closer to a design tool, as discussed below. Two others, *Wevva* and *No Second Chance*, are more focused fluidic games in specific genres, soon to be publicly released on the iOS App Store.

The basis of the Gamika Technology platform is a 2D game engine parameterised by 284 features that we have identified as core to a diverse range of casual games. This set includes parameters controlling the physics engine, player interactions and scoring/win-conditions. Physics parameters expose common features of a 2D physics engine: object spawn rates/locations, collision responses, attractive/repulsive forces, etc. Interaction parameters specify

how players interact with the physics world, such as when and how objects respond to the player tapping or dragging on the screen. Scoring and win-condition parameters specify how events impact the game outcome (the more narrowly conceived rules of the game). A more detailed parameter overview is given in (Powley et al. 2016, Section III).

Games for the Gamika platform are encoded in parameter chromosomes; the term is borrowed from evolutionary algorithms, as automated game generation is a part of each fluidic game. The chromosomes are augmented with data such as graphical and sound assets. Given a chromosome, the Gamika platform can run the game via an interpreter that allows runtime changes to the game specifications. Figure 1a shows four example Gamika games, each designed using the *Cillr* app, described in the next section.

Apps

Cillr

The first app built on the Gamika Technology platform is called *Cillr*, which enables navigation of the entire Gamika design space. Although this can be seen as a type of fluidic game, due to the size of the design space and relatively unfocused nature of the app, we use it primarily as an in-house design tool, from which we have drawn lessons used to build the more focused fluidic games discussed in the subsequent two sections, which are intended for public consumption.

Cillr implements baseline versions of both manual and automated navigation of a parametric design space. The most direct way of manually navigating the 284-dimensional design space is to give the user 284 sliders, with which they can set each parameter. While this approach – implemented in *Cillr* – is simplistic, it does work fairly effectively. The sliders are grouped into categories with related functionalities to make them more discoverable (the spawning-related sliders are collated, the collision-related sliders likewise, etc.). A few panels of the app are shown in Figure 1b.

The simplest way of automatically navigating a large parameter space is to randomise the parameters. However, we have found that this produces too low a yield of playable games, and hence *Cillr* mutates subsets of parameters from existing games instead. Randomly mutating multiple sets to produce a new random game, and then trying to figure out what it is, can be a fun interaction loop. If the user isn't interested in understanding and exploring the entire design space, however, the proportion of playable games remains too low for the mutation approach in *Cillr* to be ready for end-user consumption.

Besides producing Gamika chromosomes (both manually and with randomisation), *Cillr* includes editing tools for graphical elements such as sprites, level layout, and lighting, so complete games can be produced, including games with level progressions and multiple levels of difficulty. We have used the interface to produce clones of classic games like frogger, asteroids and space invaders, as well as a variety of novel casual games; a narrated set of design sessions is reported in (Colton et al. 2016).

As an initial baseline, *Cillr* is usable, at least by experts, though it does not yet contain interesting levels of automated

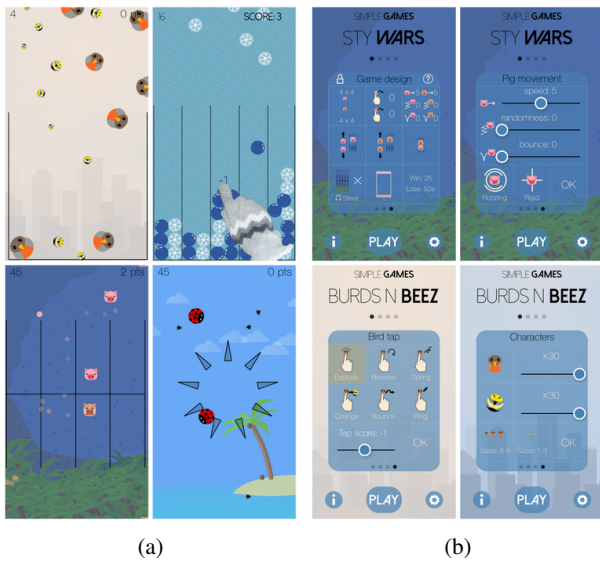


Figure 2: (a) Four *Wevva* games produced with the in-app design interface (b) which is described in the text.

game design. Its main drawback is that it is complicated to navigate, requiring some time to hunt for the correct slider to change to make something specific happen. Furthermore, even after having found the desired parameter, it can be difficult to understand why the game didn't change as expected.

In a preliminary user test with game-design undergraduate students, we found them somewhat frustrated by the experience of using *Cillr* to make games. Interface complexity was one issue, but more importantly, the difficulty of understanding the high-dimensional design space made it hard for these initial testers to grasp what they wanted to do in the app, and how they would begin to do it. Therefore, rather than focus initially on improving *Cillr*'s interface or including more automated co-design elements, we have instead focused on producing design tools for more cohesive, lower-dimensional design subspaces of the Gamika Technology platform that do not expose the entire design space at once. The first two of these are discussed below.

Wevva

Using *Cillr*, we made a four-in-a-row game called *Let It Snow*, where snow and rain pour down from the top of the screen (as white and blue balls respectively). When four or more white balls cluster together, they explode and the player gains a point for each in the cluster. Each white ball that explodes is replaced by a new one spawned at the top, with a maximum of 20 on screen at any one time. Likewise with blue balls, except the player loses one point for each that explodes. Players can interact with the game by tapping blue balls to explode them, losing a point in doing so.

While the game rules are straightforward, we have found *Let it Snow* to be challenging and require puzzle-solving strategies. There is a grid structure which collates the balls into bins, and the best way to play the game involves trap-

ping the blue balls in groups of twos and threes at the bottom, while the whites are exposed and are continually refreshed through cluster explosions. Occasionally, when all blues are trapped in small clusters, only whites will spawn, which is akin to snowing (hence the games name) and is a particularly pleasing moment to aim for.

We used *Cillr* to produce a number of variations of *Let It Snow*, initially also with the winter precipitation theme, but since expanded to include multiple settings and seasons, as well as characters such as pigs, bees and frogs. The latter were added because feedback from playtesters in our rapid game jams (described later in this paper), who had used an early version of the app lacking living characters, found it difficult to invent narratives explaining what each game was about. This expanded design space will be released as an iOS game entitled *Wevva* (Figure 2).

This app further includes two aspects that are not common in casual games: (a) an AI player that can assist novice players, and (b) a design screen enabling players to edit the games' mechanics, as well to generate levels in a semi-random way as a source of inspiration. In *Let It Snow*, the AI player appears on-screen as a gloved hand that taps the blue balls to keep clusters of four from forming (Figure 2a, top right), implementing one part of a winning strategy. A slider lets the player change the level of AI assistance. At 50%, it feels like having an in-game partner helping out. At 100%, the game is quite different, as the AI player takes care of one aspect of the game (avoiding losing points), freeing the player to concentrate on gaining points.

The design screen (Figure 2b) exposes many elements of the game design to the player: (a) what happens when the player taps on a sprite, such as exploding, changing direction or transforming into another kind of sprite (b) the shape, size and control scheme for the controller or grid (c) the sprites that exist in the game (d) scoring attached to events such as sprites exploding, being tapped, hitting screen edges or forming clusters (e) sprites' spawning locations, speeds, and limits (f) sizes of sprites (g) physics parameters, namely bounciness, wind strength and initial speeds (h) win/loss parameters, namely a time limit and score target (i) background art setting the location and (j) music selection.

There is an inspiration button designed as a brainstorming assistant, which will set these parameters in a varied way, but designed so that the clustering score mechanic is balanced in terms of expected score. We achieved this by running online simulations of novice players and recording the number of times that clusters of each size and type occurred. Finally, there is a *clean slate* feature, which resets parameters to a standard starting point.

We have conducted a series of "rapid game jams" using *Wevva*, described in the Cultural Contexts section below. These have helped to promote mixed-initiative creativity in fluidic games through events, and to refine their concept and design by observing what people do with fluidic game apps.

No Second Chance

Again using *Cillr*, we designed a game of patience and concentration, *Pendulands*. Here, balls move in a pendulum-type motion and annihilate each other if they collide; the

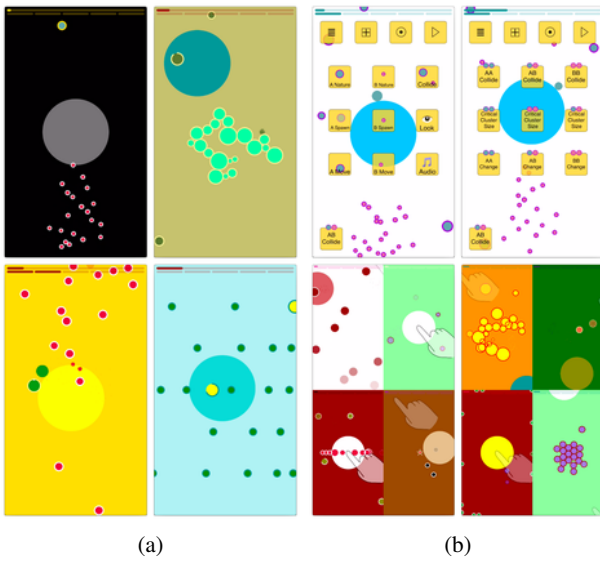


Figure 3: (a) Four *No Second Chance* games produced with (b) the in-app design interface. The top two design-interface panels show manual navigation of the space of parameters, and the bottom two show automatic game generation with split-screen auto-playtesting.

player must catch five of them by hovering under them with a large round target until they stick. By varying parameters within this theme, we discovered that a whole set of *Pendulands* variants (or levels) can be created. The anchor points defining this sub-space of Gamika games are: the player always controls the target by dragging and must catch five balls on the target. Within these parameters, very different games can be created, both in terms of their game dynamics and in terms of the types of challenges they pose.

No Second Chance is our third app, built around this space of games, a few of which are shown in Figure 3a. The name comes from a meta-game mechanic: players can send games to each other in such a way that they are deleted if the receiver doesn't beat the game on first playing (in five minutes). This emphasises the disposable nature of games in a generative space, where part of the challenge is exploring the space of games and figuring out how each one works when first encountering it.

As with *Wevva*, a design screen (Figure 3b, top) lets players make new *No Second Chance* games. It is laid out as a hierarchical menu, with submenus allowing visual style and a variety of physics parameters to be changed. Since what is fixed about *No Second Chance* games is the control and scoring mechanism, new games are made by varying physics, spawning and scoring options, which can produce very different game dynamics and mechanics. To demonstrate the types of games that can be produced (and to provide an initial challenge), the app comes with 100 games we designed using this interface, which we've categorised into three primary types of challenges: skill games, where the primary challenge is dexterity; ingenuity games, where the primary

challenge is figuring out a specific trick or strategy; and patience games, which involve waiting for the right situation to arise and capitalising on it accordingly.

The generation button creates a new game via an evolutionary process. In particular, meaningful blocks from existing games' chromosomes are crossed over, randomly mutated, and then filtered using static heuristics to reject clearly bad candidates. The first four candidates that pass the filter are auto-playtested on the device in a split-screen view (Figure 3b, bottom) that plays them at 8x speed for 5 seconds, the equivalent of 40 seconds of game time. We want games to be playable but not too easy, so the app chooses the game that the automated playtester was able to catch the most balls on, without being able to catch all five.

The split-screen visualisation of playtesting isn't strictly necessary; games could simply be silently generated and given to the user, which would be computationally cheaper as well. But this kind of "Hollywood AI" visually externalises to users what the apps' AI components are doing. It can also be entertaining in itself to watch the generation process, as new games are created and then played at rapid speed by the automatic playtester.

The term *Hollywood AI* comes from the frequent use in films of flashy computer interface mock-ups, which we use as a reference point. These are designed to look glitzy and active and to convey an idea of what the system is doing. While sometimes derided by technologists for not bearing much resemblance to real computers, the entertainment and progress-externalisation aspects of imagined film interfaces can be usefully adapted in real designs (Shedroff and Noessel 2012). They translate especially well to computational creativity systems, where it is key for the AI software to communicate when it takes the initiative and what it is doing, in this setting ideally through the use of readable visual conventions, e.g., the usage of an on-screen hand painting pictures with *The Painting Fool* (Colton and Ventura 2014).

Applications in Cultural Contexts

Rapid game jams

Game jams are events in which small teams make a game in a much shorter period of time than in traditional commercial game development. Typically lasting between 24 and 72 hours, these events bring some of the community-building and culture of LAN parties, in which players *play* games (Taylor and Witkowski 2010), to the process of game development. This produces "accelerated, constrained and opportunistic game creation events with public exposure" (Kultima 2015). They have had a large cultural impact on the indie game community, and have helped in developing a more experimental and inclusive game-development scene (Westcott 2013; Bonaiuto et al. 2014).

We aim to capture these positive aspects of game jams, but at a much shorter timescale, and with more focus on design experimentation and less on implementation. Despite being much less rigid than traditional game development, game jams still largely resemble a software hack-session, in which teams spend several days implementing an initial idea as a prototype (Musil et al. 2010). Researchers have found

that teams tend to start with an idea that remains largely intact (even if scaled down or modified where it turns out to be infeasible), with most of the time spent implementing a working prototype of that idea, rather than more free-form design experimentation (Zook and Riedl 2013).

We would like to foster game jams that emphasize design experimentation to a greater extent, with radically cut down time commitments. While making a game in a weekend is already much less of a commitment than forming a company and spending months on it, it is far more of a commitment than playing a casual game during spare time, which is our reference point for the context in which fluidic games should be both playable and designable.

Rapid game jams, lasting from as little as ten minutes up to a maximum of two hours, fit this role. In their shortest form, we have held in-office ten-minute game jams with people already generally familiar with the apps, to test and improve the ability to support this kind of tightly constrained creative design. We have also held longer 90-minute sessions (described below) with several large groups of children who had never used the apps before. This allows some time for participants to initially play the fluidic games’ built-in games to orient themselves in the design space, followed by exploring new designs through the automatic generator and/or the design interface.

The first large-scale rapid game jam we held with *Wevva* was with 65 members of Girlguiding Cornwall’s Brownie programme (i.e., girls aged 5-9), who visited Falmouth University as part of a larger Girls Can Code event on 18th February 2017. This was conducted in two sessions, with 35 users in the first and 30 in the second. Users were paired up with typically two (occasionally three) users per iPad. We began by asking them to play the included games for ten minutes. Then, we provided a brief introduction to the design interface and gave them about an hour to design their own games. This was followed by a period where they shared their games with other participants. Concluding the sessions we handed out a feedback form which contained a set of questions about their experience with the app.

The playtest of the four built-in games produced largely negative results. These games are puzzle-oriented, requiring the player to be patient and come up with a winning strategy; but here, very few were able to discover a winning strategy. The design side of the user test was more successful, however, as the children proved adept at using the built-in design tools to design other types of games in this space of games, which they preferred to the built-in games. The games they designed were generally more action-oriented, where tapping quickly on things is the winning strategy, and where there is more instant feedback about when the player took a good or bad action.

We conducted two further rapid game jams with 40 members of Girlguiding Cornwall’s Guides programme (i.e., girls aged 10-14), who visited Falmouth University on 23rd February 2017, also as part of the Girls Can Code event. We used the same approach and structure as in the first game jams, starting with an introduction to the included games and a ten-minute game playing session to familiarise them with the game, its controls and mechanics. In con-

Mechanic	Used as primary	Used at all
Herding to collect	21	39
Tap-em-up	13	42
Keeping separate	12	30
Catching to collect	8	10
Batting away	7	17
Spawning flow	6	13
Toy-like	3	10
Protect sprites	1	7
Protect zones	1	6
Steady hand	0	2
Fast reaction	0	1

Table 1: Classification of the game mechanics used in the 72 games saved by participants in the Girlguiding Cornwall rapid game jams.

trast to the younger participants from the first two sessions, the older participants spent more time playing the four included games. They also approached the four games more closely to our expectations, probing and trying out different strategies. Repeating the structure, we gave them an initial introduction to the design space, and, as with the first two groups, the possibility to explore the design space by creating and sharing their own games. Similar to the younger groups, we also concluded the sessions by handing out our feedback form which contained questions about their experience with the app. Our first observations of their exploration of the game space showed that the designed games sometimes still focused on fast tapping game mechanics, but we also saw a wider range of games which required more sophisticated strategies.

During each of these rapid game jams, we encouraged participants to save games they liked and share them with others. At the end of the sessions, we collected the saved games, totalling to 72, for analysis. Table 1 presents one way to get an overview of the design space explored in the game jams by grouping the games according to the game mechanics they use. We labeled each game with the primary game mechanic it makes use of and one or more secondary mechanics. The game mechanics we identified are:

- Herding to collect: group together clusters of sprites.
- Tap-em-up: tap as rapidly as possible on certain sprites.
- Keeping separate: keep some sprite types apart.
- Catching to collect: catch sprites with the controller.
- Batting away: knock some sprite types off the screen.
- Spawning flow: try to manipulate spawning patterns.
- Toy-like: focus on enjoyable interaction, not scoring.
- Protect sprites: keep a sprite type from exploding.
- Protect zones: keep sprites from going off screen at certain places.
- Steady hand: make careful movements, e.g. to thread through a narrow gap.
- Fast reaction: make rapid, precise movements or taps.

As can be seen in Table 1 the *herding to collect* and *tap-em-up* mechanics featured in some form in more than half of the games. This is not surprising, since scoring by forming clusters and scoring by tapping are two of the more straightforward options in the design space. The least common mechanics, seen in only three games and not in any case as the primary mechanic, were those based on skill in controlling the controller or sprites, whether the steady-hand or fast-reaction kind of skill. We had designed a number of games using these mechanics in our own 10-minute game jams, so that was an interesting difference to notice.

One aspect of automation our current apps do not have is automatic fix-up of games to balance them and avoid exploits, although we have done research on a version of automatic tweaking that runs server-side (Powley et al. 2016). To see whether such a feature would be important to add, we classified the 72 games according to whether we, as expert players, were able to quickly find an easy exploit in the game design. We were able to do so in 31 of the games. Of these, the two most common exploits were being able to win by indiscriminately tapping (seen in 22 games) and being able to win by doing nothing at all for a short period of time and win (found in 7 games). On the other hand, since *tap-em-up* games were one of the two most common mechanics used, it's not clear that winning by indiscriminate tapping would actually be considered an exploit by the designers. We observed, for example, some pairs of users sharing an iPad taking turns playing a very easy game requiring rapid tapping, but competing to beat each others' best scores.

Besides this analysis of game mechanics and exploits, we asked the game-jam participants to fill out a survey about their experiences. We have performed a preliminary analysis of these survey results, for 30 girls of average age 12, responding to survey questions quantitatively using the visual analog scale. While a full user-study analysis is beyond the scope of this paper (which focuses on cultural applications of fluidic games), we have found a few interesting results so far. The following are four statistically significant correlations we observed:

- Positive: Between using the inspiration button and finding games produced by the inspiration button useful.
- Negative: Between interest in a career in game design and using the clean-slate restart.
- Positive: Between interest in a career in game design and enjoying using the app.
- Positive: Between playing a lot of games and enjoying using the app, as well as feeling more creative.

Wevva therefore seems to attract people who tend to frequently play games; those frequent players also feel more creative using the app than novice users. Those who are interested in becoming game designers used the clean slate less as they seem to feel that the games they produce from that point in the fluidic space are less interesting. Those not interested in a career in game design explore more around the clean slate, perhaps because it gives a familiar starting point for exploring the game space.

Classroom usage

Through a collaboration with a local school, the Camborne Science and International Academy, we have been developing fluidic games into a curriculum suitable for use in classrooms. The first version of this curriculum, currently in progress, teaches game design through weekly sessions of 90 minutes each, designed around experimentation in *No Second Chance*. Each week's lesson introduces a new aspect of game design, and most lessons also use that element of game design as a hook through which to teach material from another relevant subject. For example, the lesson on game visual aesthetics introduces students to colour theory, the lesson on object movement and collision response also teaches elementary physics, and the lesson on the included automated game generator introduces students to artificial intelligence and Computational Creativity.

Use game design to organise a technology-based curriculum is similar in some respects to curricula that introduce programming in schools through visual programming languages such as Scratch (Resnick et al. 2009) or its tablet version ScratchJr (Strawhacker et al. 2015), which often use games as the motivating example that shows students what can be done if one learns to code. While we draw inspiration from these projects, our goal is to focus less on teaching *programming* specifically, and more on teaching *design* in a computational setting, emphasising that programming, while an important skill, is not the only aspect of game design, nor of computational thinking more generally. The lessened focus on coding as the specific skill to teach also frees up a larger part of the curriculum to focus on the connections to other fields, such as the colour-theory and physics examples mentioned above.

Since we are currently part of the way through the initial pilot of this curriculum based on fluidic games, we report only preliminary observations. Game design in *No Second Chance* is essentially exploring a physics-based game space through parameters that produce new types of gameplay dynamics, so the curriculum is based around introducing parts of the parameter space each week, explaining what these parameters mean, how to design in that space, and how the new set of parameters impacts game design. The exercises were designed in a way to incrementally build up knowledge about the physics and parameter space of *No Second Chance*, starting with basic control over when and where objects appear on screen to how to use more sophisticated combinations of parameters to balance designed games. Each lesson sheet contains a set of parameters explained in detail with examples to guide further exploration. As proposed by (Resnick 2004), the usage of games should still be playful, so instead of giving them a fixed set of provable exercises, each is focused on open-ended exploration of the design space, with a soft peer evaluation/assessment based around sharing and critiquing each others' games (an activity that also, through playing others' games, helps students notice aspects of the design space they may have missed).

Structuring an introduction to *No Second Chance* as a series of lessons also helped us to better understand its design space as a fluidic game. To aid the students with their exploration of the game space and at the same time teach them

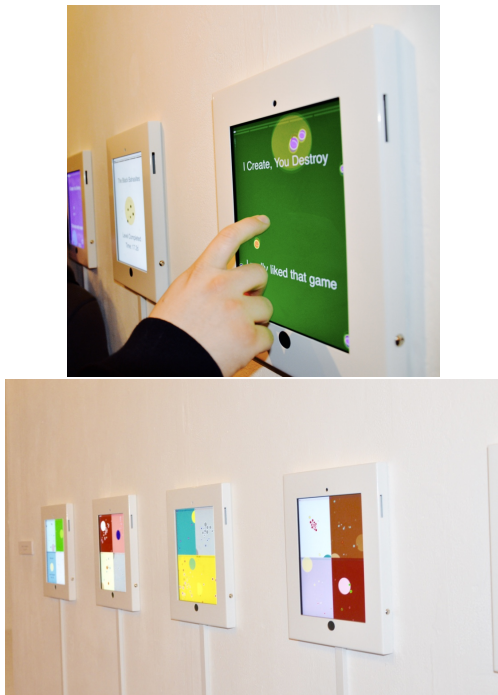


Figure 4: The *I Create, You Destroy* interactive installation. Top: An audience member destroys a generated game just as the AI was playing it. Bottom: An array of iPads happily generates and automatically playtests new games.

a further lesson each week (such as elementary physics), forced us to critically think about sets of parameters and their relation to each other and how this can be used to conceive of the parameter space in a more structured manner. While observing students' exploration, we also became aware of potentially new and interesting areas of the design space we have not taken into account yet. The amount of interest we saw during those sessions both motivated us and demonstrated the potential of employing more explorative and creative games into teaching and learning.

An art installation

I Create, You Destroy is an installation of six iPads created by @ThoseMetaMakers, the art collective alter ego of The MetaMakers Institute. This first exhibition by @ThoseMetaMakers was shown at the first Games As Arts / Arts As Games festival, which was held at The Poly in Falmouth, Cornwall, from the 12th to 22nd October 2016.¹

In this work, an autonomous version of *No Second Chance* creates, plays, critiques, selects, and shares abstract artistic games. It happily makes and plays games all day long, alternating between generating games, using the split-screen rapid-fire auto-playtesting of the regular *No Second Chance* app, followed by playing the generated game at normal speed. It continues in this generate-and-play loop unless a visitor touches one of the screens, in which case the game

¹metamakersinstitute.com/gamesasarts

that was being made or played is destroyed forever, and the visitor is told this (Figure 4). Referencing and challenging the long-standing but highly topical worries that machines will take jobs and pose existential threats, *I Create, You Destroy* is an artificial intelligence system, but it is not the bad guy in this case.

In aiming to challenge mainstream assumptions about AI being a threat, the concept of the piece resides in the balance produced by using a mainstream platform (iPad) and a mainstream genre (casual games) to challenge this mainstream assumption. The AI hides behind the bold, sharp and colourful surface of six iPads fixed to the white background wall. With white cases and white background, only the playful images of Gamika stand out. The small balls move fast, appearing from everywhere, going anywhere. New games are created directly in front of visitors, who are faced with a choice to make: either merely watch, or act, interact and then themselves become a threat to creativity.

Breaking the assumption and questioning what we are expected to do is the core of *I Create, You Destroy*, as the audience is presented with a tablet, which they are normally expected to touch, running a game, which they are normally supposed to play. However, this is not what they should do in this case; in moral terms the audience acts destructively if they fulfill the usual expectations of interaction.

The exhibition included works from a number of other artists working with games as a medium, such as Alan Meades, Ian Gouldstone and Oliver Sutherland; as well as artwork from games such as *Lumino City* (State of Play) and *Machinarium* (Amanita Design). One point of connection, where *I Create, You Destroy* meets Ian Gouldstone's piece *Cruise Control 2020* and Oliver Sutherland's *Untitled (Loosing it)*, is the idea of the continual moving image. In the tradition of art built on game technologies (Bittanti and Quaranta 2009; Sharp 2012), these works play with game tropes, but are not presented as games the viewer themselves can play. And yet the spectator is not merely watching a pre-recorded movie, when the creative process is happening right there. What is going on will never occur again. We could consider this uniqueness one of the holy quests of contemporary art and Computational Creativity. Since we lost the unique nature of the artefact, we seek the exceptional character of the experience. Games technologies and AI enable the rise of a new hybrid art form between installation and performance with something sculptural and cinematographic about it.

Conclusion

We have introduced and investigated fluidic games, which are casual creator apps for hand-held devices. Fluidic games are in the genre of *maker-games*, games that can be modified by users in various ways. However, with fluidic games, users can not only design levels or skin games, as is common in other maker-games, but can change underlying game mechanics such as the physics, scoring mechanisms, player input, and how objects interact.

In contrast with game-creation environments such as Scratch Junior, game design in fluidic games can be achieved without any coding requirements. Indeed, we have designed

our fluidic games to have fun, efficient user interfaces, so that the line between making and playing a game is fairly blurred. To achieve this has required some element of Computational Creativity support in the app, and we have described how automatic game generation and auto-playtesting have enabled users to search larger spaces of the game space, with less frustration than they would do without these tools.

Since game-playing and game-making are always situated in cultural contexts, we have been experimenting with how fluidic games fit in three cultural contexts, in order to drive both our design and technology development through real-world experience. Our three pilot settings are: 1) using fluidic games to host rapid game jams, which last no more than 1-2 hours and focus more on exploring design possibilities than game implementation, 2) integrating fluidic games into a school curriculum in order to both teach game design, and use game design as a hook with which to introduce concepts such as colour theory, physics, and artificial intelligence, and 3) adapting an autonomous version of fluidic games as an art installation, with an AI agent both designing and playing the games, in this case to comment on the assumption that AI is likely to play destructive, dangerous roles in society.

Open technical research problems include: improving automated game generation and automated game playing in open-ended spaces, enabling more close coupling between the generative mode (currently used for brainstorming) and the user-operated design interfaces, and developing new methods to enable users to playfully explore design spaces (the latter includes developing methods to better familiarise users with the Computational Creativity concept of design-space navigation in the first place).

Acknowledgments

This work is funded by EC FP7 grant 621403 (ERA Chair: Games Research Opportunities). We are grateful to Girlguiding Cornwall and the Camborne Science and International Academy for their collaboration.

References

- Bittanti, M., and Quaranta, D. 2009. *Gamescenes: Art in the Age of Videogames*. Johan & Levi Editore.
- Bonaiuto, A.; Mingrino, M.; Sampugnaro, R.; Fallica, S.; and Mica, S. 2014. Participation at the Global Game Jam: A bridge between consumer and producer worlds in digital entertainment. *GAME* 3(2):35–45.
- Colton, S., and Ventura, D. 2014. You can't know my mind: A festival of computational creativity. In *Proc. Intl. Conference on Computational Creativity*.
- Colton, S.; Nelson, M. J.; Saunders, R.; Powley, E. J.; Gaudl, S. E.; and Cook, M. 2016. Towards a computational reading of emergence in experimental game design. In *Proc. Computational Creativity and Games Workshop*.
- Compton, K., and Mateas, M. 2015. Casual creators. In *Proc. Intl. Conference on Computational Creativity*.
- Cook, M.; Colton, S.; and Gow, J. 2016. The ANGELINA videogame design system. *IEEE Transactions on Computational Intelligence and AI in Games*.
- Grace, K., and Maher, M. L. 2014. Towards computational co-creation in modding communities. In *Proc. Workshop on Experimental Artificial Intelligence in Games*, 15–20.
- Kultima, A. 2015. Defining game jam. In *Proc. Conference on the Foundations of Digital Games*.
- Liapis, A.; Smith, G.; and Shaker, N. 2016. Mixed-initiative content creation. In *Procedural Content Generation in Games*. Springer. 195–214.
- Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Computational game creativity. In *Proc. Intl. Conference on Computational Creativity*.
- Musil, J.; Schweda, A.; Winkler, D.; and Biffel, S. 2010. Synthesized essence: What game jams teach about prototyping of new software products. In *Proc. Intl. Conference on Software Engineering*, 183–186.
- Nelson, M. J.; Colton, S.; Powley, E. J.; Gaudl, S. E.; et al. 2017. Mixed-initiative approaches to on-device mobile game design. In *Proc. CHI Workshop on Mixed-Initiative Creative Interfaces*.
- Powley, E. J.; Colton, S.; Gaudl, S. E.; Saunders, R.; and Nelson, M. J. 2016. Semi-automated level design via auto-playtesting for handheld casual game creation. In *Proc. IEEE Conference on Computational Intelligence and Games*, 372–379.
- Resnick, M.; Maloney, J.; Monroy-Hernández, A.; et al. 2009. Scratch: Programming for all. *Communications of the ACM* 52(11):60–67.
- Resnick, M. 2004. Edutainment? No thanks. I prefer playful learning. *Associazione Civita Report on Edutainment* 14.
- Sharp, J. 2012. A curiously short history of game art. In *Proc. Conference on the Foundations of Digital Games*.
- Shedroff, N., and Noessel, C. 2012. *Make It So: Interaction Design Lessons from Science Fiction*. Rosenfeld Media.
- Shneiderman, B. 2007. Creativity support tools: Accelerating discovery and innovation. *Communications of the ACM* 50(12):20–32.
- Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):201–215.
- Strawhacker, A.; Lee, M.; Caine, C.; and Bers, M. 2015. ScratchJr demo: A coding language for kindergarten. In *Proc. Intl. Conference on Interaction Design and Children*.
- Taylor, T. L., and Witkowski, E. 2010. This is how we play it: What a mega-LAN can teach us about games. In *Proc. Conference on the Foundations of Digital Games*, 195–202.
- Westecott, E. 2013. Independent game development as craft. *Loading...* 7(11).
- Yannakakis, G. N.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-initiative co-creativity. In *Proc. Conference on the Foundations of Digital Games*.
- Zook, A., and Riedl, M. O. 2013. Game conceptualization and development processes in the Global Game Jam. In *Proc. FDG Workshop on the Global Game Jam*.

Incorporating novelty, meaning, reaction and craft into computational poetry: a negative experimental result

Carolyn Lamb, Daniel G. Brown, and Charles L.A. Clarke

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Abstract

We present TwitSonnet, a Twitter found poetry system. TwitSonnet attempts to build meaningful poems based on criteria we previously identified as separating good computer-generated poems from bad ones: namely, novelty, meaning, reaction and craft. We show the results of an experiment with human raters that shows that TwitSonnet poems focusing on these criteria are not artistically superior to poems that do not. We discuss the implications of this negative result for TwitSonnet's development, and the general implication of negative experimental results on computational creativity as a field.

Introduction

Computational poetry is a popular area of computational creativity in which computers are programmed construct poems. A variety of approaches have been used for this construction; a summary of the different approaches and their similarities and differences can be found in our previous paper (Lamb, Brown, and Clarke 2016b). These approaches range from simple word substitutions to very sophisticated systems using neural networks to replicate patterns in human poetic language.

One of the many existing approaches to computational poetry is found poetry, in which a computer selects appropriate excerpts of human-generated texts and remixes them into a new poetic work. A few systems, including Ranjit Bhatnagar's Pentametrion (Bhatnagar 2012) and Andrei Gheorge's The Longest Poem In the World (Gheorge 2009), generate found poetry by taking text from Twitter. These systems are simplistic, choosing tweets based only on rhyme and number of syllables. Hartlová's Mobtwit (Hartlová and Nack 2013) performs a more sophisticated analysis, creating limericks out of tweets chosen for emotional contrast. However, systematic or falsifiable analysis of what makes a tweet suitable for use in found poetry has not yet been done.

TwitSonnet

TwitSonnet (Lamb, Brown, and Clarke 2015b) is a found poetry system similar to Pentametrion (Bhatnagar 2012). Both systems assemble pairs of rhyming, 10 or 11-syllable tweets into a sonnet. Where TwitSonnet differs from Pentametrion is that we try to select tweets that are, in ways we will define

below, more poetic than others. The hope is that a Twitter sonnet containing more poetic lines will be more meaningful, more entertaining, and potentially more creative than a sonnet containing only arbitrary tweets.

The ability to evaluate unfinished work - including the suitability or unsuitability of potential components of the work - is a vital part of the creative process (Galanter 2012). Throughout TwitSonnet's development, our goal has been to focus on automating this relatively high-level judgment while abstracting away from the low-level generation of language. A system that can be shown to intelligently select lines from a corpus can then be trusted to use intelligent selection on lines of its own.

How TwitSonnet works

TwitSonnet creates topical poems out of tweets using the following stages:

1. **Data gathering.** We use the Tweet Archivist service (Tweet Archivist 2016) to pick tweets containing a topical keyword during an appropriate time interval.
2. **Filtering.** TwitSonnet counts the syllables of the tweets in the gathered data and groups them by end rhyme, using a modified version of the code from Hirjee and Brown's Rhyme Analyzer (Hirjee and Brown 2010). This code, built on top of the CMU Pronouncing Dictionary (Weide 1998), allows for imperfect rhymes with an adjustable threshold. Any tweet which has fewer than the appropriate number of syllables for a sonnet or which does not fit into a rhyme grouping with at least one other tweet is discarded. Tweets with more than the appropriate number of syllables are split into their constituent sentences, if possible. Excessive hashtags and other unpronounceable features of tweets are also removed. Because of these methods, some lines in the rhyme groupings do not contain the original keyword, but are potentially related due to their proximity to the keyword in their original context. Our modified Rhyme Analyzer code can appropriately handle common forms of Twitter slang and misspellings, but discards tweets that contain obvious non-words or are not in English.
3. **Ranking.** Tweets are given scores for desired poetic criteria as described below. Scores are normalized by range

and then the different scores for each tweet are added together.

4. **Selection.** The seven rhyming pairs of tweets with the highest scores (judged based on the second-highest tweet in the rhyming set) are selected to be placed in a sonnet.
5. **Reordering.** Optionally, the selected lines can be re-ranked and placed in a meaningful order. For example, they could be ordered from the most abstract introductory statements (least imagery) to the strongest concluding image (most imagery). Otherwise, the tweets are ordered according to score, with the highest scoring couplet at the end.

TwitSonnet is a fully functional system which can create a sonnet out of any sufficiently large collection of tweets. From July through the end of October 2016, we posted several of TwitSonnet's poems per week at <http://twitsonnet.tumblr.com/>.

Poetic criteria

There are various ways to evaluate the success of a creative computer system. For this project, we are focusing on the Product perspective (Jordanous 2015) in which the system is primarily judged on the quality of its output. But how, specifically, do we define quality? In previous work (Lamb, Brown, and Clarke 2016a), we developed a set of domain-specific product-based criteria for computer-generated poems by studying the responses of Experimental Digital Media graduate students to a varied and inclusive set of such poems. We grouped the desired traits expressed in these students' responses into four categories:

- **Reaction** - the reader sees the poem as interesting, or has an emotional response, based on their prior experience of poetry.
- **Meaning** - the poem coherently expresses an idea.
- **Novelty** - the poem is new, different, or subversive.
- **Craft** - the poem is written skillfully, with good use of form (if any), imagery, and poetic devices.

These categories bear parallels to, but are distinct from, other existing formalizations for evaluating digital poetry. Criteria similar to Craft, for example, appear in van der Velde's creativity criteria (van der Velde et al. 2015), in Manurung et al.'s domain-specific criteria for computational poetry (Manurung, Ritchie, and Thompson 2012), and in the Creative Tripod model (Colton 2008). A more detailed comparison of our categories to other models appears in our previous study (Lamb, Brown, and Clarke 2016a).

We developed TwitSonnet's algorithm specifically taking these categories into account, as follows:

For **reaction**, we gave a higher score to tweets containing words pertinent to a desired emotion, as measured by the NRC Hashtag Emotion Lexicon, which was created specifically for Twitter (Mohammad and Kiritchenko 2015). The Emotion Lexicon contains eight different emotions. We chose a desired emotion for each poem by measuring which emotions were most prevalent in the gathered data, and then

normalizing by the rate of emotion words in non-topical data.

For **meaning**, we chose tweets relevant to a specific topic through a two-step process. First, the data gathering process using Tweet Archivist narrows in on a topic by selecting tweets by time range and keyword. Second, at the ranking stage, we created a trigram frequency data set for the tweet corpus and gave higher scores to tweets consisting of trigrams with high frequency scores.

For **craft**, we did two things. First, as mentioned, we selected tweets for rhyme and meter and arranged them into a sonnet, which is a recognized poetic form. Second, we gave a higher score to tweets containing stronger primary process imagery, as measured using the Regressive Imagery Dictionary (Provalis 1990). The Regressive Imagery Dictionary gives higher scores to "primary process" words relating to physical senses, experiences, drives, and the body, and lower scores to more abstract, "secondary process" words. Selection for such concrete physical imagery in poetry is supported by Simonton (Simonton 1990), who used the Regressive Imagery Dictionary to show a greater presence primary process imagery in more successful sonnets, and by Kao and Jurafsky (2012), who used related measures to show that professional contemporary poetry uses more concrete imagery than the poetry of amateurs.

For the purposes of this study, we did not find a satisfactory method of measuring novelty. Some obvious attempts, such as selecting for unusual trigrams, seemed to only increase the number of off-topic, "random", and nonsensical tweets. In context of our previous study, the category of Novelty refers to interesting juxtapositions, new thoughts, and subversions of existing concepts, not to this type of "mere novelty". We did reduce repetitiveness by placing a limit on the number of times TwitSonnet was allowed to repeat a poem's keywords, replacing repetitive tweets with the highest ranked alternatives that did not contain the topic keywords.

These specific operationalizations of our criteria are made for the specific domain of found poetry, and the criteria would be operationalized differently in a poetry generator which was creating its text from scratch or through a template.

In summary, our system is explicitly built to satisfy our domain-specific product-based criteria. However, like any system, its success at satisfying them in practice needs to be tested empirically. We will now describe how we have tested previous and current versions of TwitSonnet.

Previous evaluation

A proof-of-concept version of TwitSonnet, then called TwitSong, was evaluated using a pair preference study. Non-expert participants compared TwitSong's poems to a control group in which the ranking stage assigned every tweet the same score (Lamb, Brown, and Clarke 2015b). Participants significantly preferred sonnets in which the ranking was based on certain criteria, especially topicality (the equivalent of our current category Meaning), to control sonnets. However, the scoring in this early study was not done by TwitSonnet, but by workers on a crowdsourcing website.

Review coming tomorrow/this afternoon.
 doctor strange was amazing. cant wait for Thor.
 closer look at the evolved hero costume
 Visually stunning, left me wanting more
 What is a Doctor Strange collector corps box?
 Check out the latest new movie details!
 So excited to see Marvel in the parks!
 what was your first Doctor Strange comic? #Strange-
 Tales
 I have 10 more tickets to give away
 Doctor Strange 8:45 Ill be there
 Doctor Strange is pretty, and pretty OK:
 gonna lowkey fall asleep in this chair
 It better be worth slacking on my dreams!
 Doctor Strange (with Christy at Platinum Screens

Figure 1: A sample of TwitSonnet’s output, regarding the movie “Doctor Strange”. (The keyphrase used was “Doctor Strange”, and the time range used was the movie’s opening weekend.)

The purpose of the study was to show that line selection based on criteria does, in fact, produce a better poem than arbitrary line selection. We then moved on to the current step of having TwitSonnet do its own, automated line selection.

Evaluating TwitSonnet

We had two goals in evaluating the current version TwitSonnet. First, we wanted to confirm that the effect of the automated scoring was similar to the effect of the crowdsourced scoring. Second, we wanted to improve on the methodology of the previous study by including expert raters, who are more consistent when rating creative artifacts than non-experts (Kaufman et al. 2008). Indeed, in the domain of poetry, judges with little to no poetry experience can have the opposite of the preferences of an expert (Lamb, Brown, and Clarke 2015a).

Method

Experts in poetry can be difficult to recruit for studies. We recruited participants using snowball sampling on the social networks of all three of this paper’s authors, particularly the first author, who is a published poet under a pen name.

Participants were asked demographic questions and classified as experts or non-experts. In keeping with the recommendations of Kaufman et al (2008), we based our definition of expertise not in the study of poetry but in experience actively generating successful poetry. Participants who had published poetry in a magazine or collection, read their own poetry at a reading or slam, and/or published digital poetry were considered poetry experts.

The poetry experts consisted of 13 women, 12 men, and 11 non-binary gendered poets. (While this is a serious overrepresentation of non-binary poets - likely an artifact of the snowball sampling method - we do not expect it to skew our

results, as none of the poems in the study pertain to gender or queer/trans* issues.) The median age was 32, ranging from 17 to 56. All but two of the experts were native speakers of English.

The non-experts consisted of 12 women, 19 men, three non-binary, and one non-expert who did not disclose their gender. The median age was 36, ranging from to 21 to 70. 29 of the 35 non-experts were native speakers of English.

As a result of our snowball sampling, most of our “non-expert” participants could actually be considered quasi-experts: they reported that they were regular readers of poetry, had written unpublished poetry for pleasure, taken classes in poetry, listened to poetry podcasts, attended poetry readings, or taught poetry to K-12 students. (An additional form of experience, being a poetry editor for a magazine or other publication, did not appear among non-experts. Seven of our 36 expert participants reported having been a poetry editor.) Only three participants had no significant experience with poetry, and one of these was a graduate of a prose creative writing program. Thus, we would expect less difference between the experts and non-experts in this study than we would see if the non-experts were completely inexperienced.

Each participant was shown 8 poems in a random order, from the same selection of 8 current events topics and 8 emotions. The topics included three topics from recent movies and television, two astronomy topics, a ban on the “burkini” in France, and two topics relevant to the recent 2016 Summer Olympics. Each topic was associated with an emotion from the NRC Hashtag Emotion Lexicon: anger, anticipation, disgust, fear, joy, sadness, surprise, or trust.

Each of these 8 poems was in turn drawn at random from one of three groups. In Group A, poems were generated using steps 1 and 2 from the TwitSonnet process, but not the remaining steps. In other words, these were our control poems, in which no filtering or reordering based on our four criteria was performed. Poems in Group B were generated using steps 1 through 4 (so they were generated and filtered using our four criteria, but not reordered), and poems in Group C used all five steps including reordering. For each of the 8 poems, participants were then asked the following questions, each on a 5-point Likert scale:

1. “How much do you like this poem?” (*Reaction*)
2. “How creative is this poem?”
3. “How well does this poem express the emotion of [emotion]?” (*Reaction*)
4. “How meaningfully does this poem summarize its topic?” (*Meaning*)
5. “How new and different is this poem?” (*Novelty*)
6. “How successful is the imagery in this poem?” (*Craft*)
7. “How cohesive is the narrative of this poem?” (*Meaning*)

The answers provided at each point of the Likert scale were

- Not at all
- Not much

- A little
- Somewhat
- Very much

Apart from “How creative is this poem?”—an irresistible option in a computational creativity project—each of the questions is designed specifically to assess TwitSonnet’s success at one of our four domain-specific categories. Our hypothesis was that the poems from Groups B and C would score higher than Group A on at least some questions, and that Group C would score higher than Group B specifically for narrative cohesion. Participants were also given a freeform text box in which to write any other comments they had about the poems.

Results

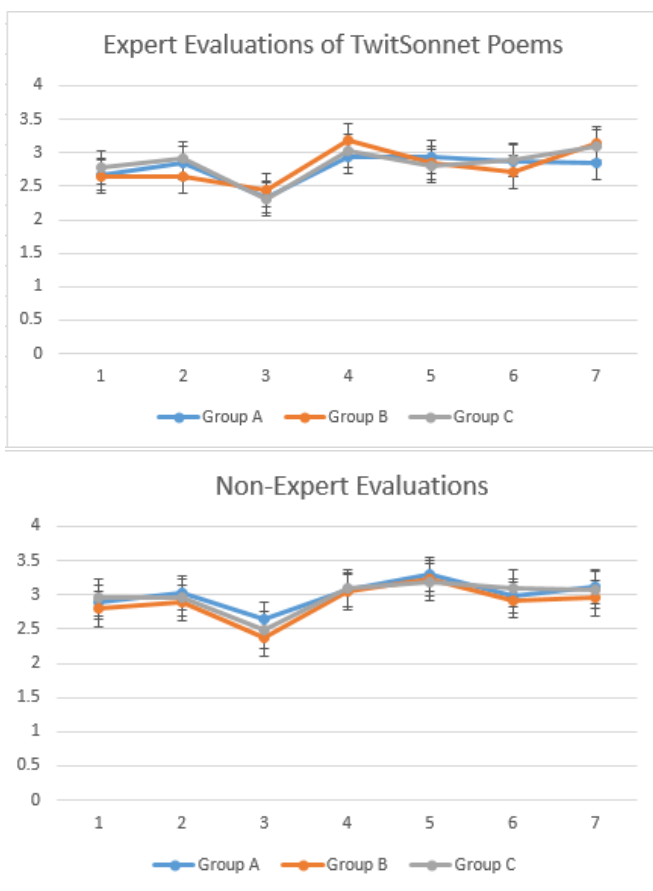


Figure 2: Experts’ and non-experts’ evaluations of TwitSonnet’s poems. The X-axis shows the seven evaluation questions in the same order as they are listed in our Method section. The Y-axis shows answers on a 5-point Likert scale, with 5 being the most positive response and 1 the least positive. Error bars show 95% confidence intervals.

Unfortunately, our hypotheses were not confirmed. As can be seen in Figure 2, there was little difference in either

experts’ or non-experts’ reactions to poems from the different groups. Standard deviations within groups far exceeded the difference in mean between groups, and for most criteria, the size of the 95% confidence interval also exceeded this difference. The largest apparent difference was in narrative cohesion as judged by experts, in which groups B and C ($\bar{x} = 3.13$ and 3.01 , respectively) outperformed group A ($\bar{x} = 2.84$)—but the standard deviations of these groups on this question were 1.29, 1.27, and 1.28, more than five times the size of this difference. None of the differences were statistically significant.

We compiled the most common freeform comments by experts and nonexperts. Experts stated that the poems seemed random and choppy; often there would be small sections with a satisfying juxtaposition but they would be mixed with other lines that didn’t fit. There was too much focus on rhyme and meter at the expense of content, with several experts stating they would have preferred if the poems did not rhyme. There were also too many lines that trailed off in the middle of a sentence or even a word. However, several experts said that they found the idea behind the project very interesting in spite of any criticism they might have of the poems. Nonexperts had fewer comments and responded more to surface features of the poem: for example, several nonexperts said they would have preferred not to see hashtags in the poems, as well as typos, bad punctuation, and other errors. Nonexperts also agreed with experts that the poems lacked coherence.

Discussion

The negative result here is surprising because, in our previous study, the difference between the equivalents of Group A and Group B was statistically significant (Lamb, Brown, and Clarke 2015b). There are three possible explanations for this.

First, perhaps the difference is due to a difference in how we performed the evaluation this time (for example, Likert scales vs pairwise preferences). While this is important to consider, we believe that, with the possible exception of narrative cohesion measured by experts, the current study shows a striking lack of difference between groups, which is not attributable merely to the use of a less sensitive statistical method.

A related suggestion is that perhaps the current events topics chosen in this study were not the correct choices. For instance, raters might have had stronger opinions about the emotions expressed in a poem if the poem was on a more polarizing topic. Such polarizing topics are plentiful in current events, especially as the study was run during the lead-up to the divisive 2016 U.S. presidential election. TwitSonnet’s online incarnation did indeed create poems on divisive political topics: an example is shown in Figure 3. We chose not to include these poems in the study so as not to conflate a rater’s political opinion with their artistic opinion of this poem. This may or may not have been the correct choice.

Second, perhaps the automated judgments we are using contain too much error when compared to human judgment and are thus not suitable for this purpose. We have deliberately used computationally simple methods in order to pro-

Final Presidential Debate (10/19)
 Donald Trump is master of the head fake
 This East Texas pole shows a leftward lean
 but goodnight this all debate gave me headache
 Much smarter than his brother Crooked John
 An interesting debate is taking place
 While America tuned in to watch Don
 Trump doing the deniro mobster face
 started by her very sleazy campaign
 Debate Watch Party SAC 305
 Donald Trump is LITERALLY insane
 watching guy fieris diners drive-ins and dives
 That was the sound of women everywhere
 Its a humanitarian nightmare

Figure 3: A TwitSonnet poem posted online, using the keyword “debate”, immediately after the 2016 U.S. presidential election debates.

cess large numbers of tweets on a large number of topics. It is possible that these methods are simply not up to the tasks assigned them.

Third, while the focus on this study was on the ranking and ordering steps, the filtering step has also improved since the previous study. Humans are unlikely to judge nonsensical tweets as being very topical or as having a clear emotion. Automated judgment is less sensitive to nonsense, and in addition, our filtering step has improved at automatically removing nonsense from both ranked and unranked poems. Thus, it is possible that some of the effect in the previous study was due the ranking step reducing nonsensical tweets, and that this reduction is no longer noticeable in the current study.

Filtering for rhyme and meter (craft) and the use of keywords in data selection (topicality) was already in place in very close to its current form in the previous TwitSong study, so these steps alone cannot be used to account for our current results, but it should be noted that due to these techniques, even the poems in Group A are not “raw” control poems in the sense of having no attention paid to the four criteria. Neither would, for example, Pentametrone’s poetry, since it too is selected for rhyme and meter (Bhatnagar 2012). The use of a *pure* control group - for example, a completely random selection of English-language tweets - would likely produce something closer to a significant result. However, it would not tell us if our filtering techniques, specifically, were working as intended.

Pearce *et al.* (2002) and, more recently, Bown (2014) call attention to the need for falsifiability in computational creativity evaluation. Unfortunately, the use of falsifiable techniques will sometimes produce a negative result. A negative result does not necessarily invalidate the worth of the project, but it is a sign that the creative system in its current form is not performing as intended.

There are several possible responses to this specific negative result. First, we could try performing a different eval-

uation. Second, we could modify our line selection techniques and engage in further analysis of existing poems to see which techniques might be most promising.

Third, we could step back and ask ourselves what goals we are working towards with TwitSonnet. A different methodology might serve those goals better. For example, if our goal is to teach a computer to identify poetic lines, we might consider using source text richer in poetic style and technique than Twitter. If our goal is to entertain with amusing poetic summaries of news events, we might ask if the present project is the best way to do that. In particular, it is notable that in both this and the previous study, Twitter’s informality and conventions such as hashtags were offputting to many participants. These may be aspects of Twitter which make it inherently more difficult as a repository for poetic speech. To verify this interpretation, one option would be to “clean” gathered tweets of hashtags, typos, and other traits that bothered the non-expert raters, before running the study again.

In all cases, a negative result like this one points to a need to reassess and change some aspects of our project, to a greater or lesser degree, so that it fits more precisely with our actual research goals.

Conclusion

While negative results can be discouraging, this result gives us information which is useful for the further development of TwitSonnet and related projects, and which we might not have obtained if we had not performed a falsifiable evaluation. We learned in the previous study that selection of tweets based on specific criteria can indeed produce superior poetry to arbitrary selection. However, we could not show using falsifiable methods that the current method of tweet selection achieved this. We have therefore learned that we should be more careful in the future about exactly how lines for a found poem are selected and what, if anything, this selection contributes to the output. As always, empirical testing is needed so as to ensure that tweet selection, or any other component of a creative system’s process, works as intended.

References

- [2012] Bhatnagar, R. 2012. Pentametrone. <http://pentametrone.com/>.
- [2014] Bown, O. 2014. Empirically grounding the evaluation of creative systems: incorporating interaction design. In *Proceedings of the Fifth International Conference on Computational Creativity*, 112–119.
- [2008] Colton, S. 2008. Creativity versus the perception of creativity in computational systems. In *AAAI spring symposium: creative intelligent systems*, volume 8.
- [2012] Galanter, P. 2012. Computational aesthetic evaluation: past and future. In *Computers and Creativity*. Berlin: Springer. 255–293.
- [2009] Gheorge, A. 2009. The longest poem in the world. <http://www.longestpoemintheworld.com/>.

- [2013] Hartlová, E., and Nack, F. 2013. Mobile social poetry with tweets.
- [2010] Hirjee, H., and Brown, D. G. 2010. Rhyme analyzer: An analysis tool for rap lyrics. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*.
- [2015] Jordanous, A. 2015. Four perspectives on computational creativity. In *The AISB15's 2nd International Symposium on Computational Creativity (CC2015)*, 16.
- [2012] Kao, J., and Jurafsky, D. 2012. A computational analysis of style, affect, and imagery in contemporary poetry. In *NAACL Workshop on Computational Linguistics for Literature*, 8–17.
- [2008] Kaufman, J. C.; Baer, J.; Cole, J. C.; and Sexton, J. D. 2008. A comparison of expert and nonexpert raters using the consensual assessment technique. *Creativity Research Journal* 20(2):171–178.
- [2015a] Lamb, C.; Brown, D. G.; and Clarke, C. L. 2015a. Human competence in creativity evaluation. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 102.
- [2015b] Lamb, C. E.; Brown, D. G.; and Clarke, C. L. 2015b. Can human assistance improve a computational poet? *Proceedings of Bridges 2015: Mathematics, Music, Art, Architecture, Culture* 37–44.
- [2016a] Lamb, C.; Brown, D. G.; and Clarke, C. L. 2016a. Evaluating digital poetry: Insights from the cat. In *Seventh International Conference on Computational Creativity*.
- [2016b] Lamb, C.; Brown, D. G.; and Clarke, C. L. 2016b. A taxonomy of generative poetry techniques. In *Proceedings of Bridges 2016: Mathematics, Music, Art, Architecture, Culture*.
- [2012] Manurung, R.; Ritchie, G.; and Thompson, H. 2012. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence* 24(1):43–64.
- [2015] Mohammad, S. M., and Kiritchenko, S. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence* 31(2):301–326. Full lexicon available at <http://saifmohammad.com/WebPages/lexicons.html>.
- [2002] Pearce, M.; Meredith, D.; and Wiggins, G. 2002. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae* 6(2):119–147.
- [1990] Provalis. 1990. Regressive imagery dictionary. <https://provalisresearch.com/products/content-analysis-software/wordstat-dictionary/regressive-imagery-dictionary/>.
- [1990] Simonton, D. K. 1990. Lexical choices and aesthetic success: A computer content analysis of 154 shakespeare sonnets. *Computers and the Humanities* 24(4):251–264.
- [2016] Tweet Archivist. 2016. Tweet archivist. <http://tweetarchivist.com/>.
- [2015] van der Velde, F.; Wolf, R. A.; Schmettow, M.; and Nazareth, D. S. 2015. A semantic map for evaluating creativity. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 94.
- [1998] Weide, R. L. 1998. The cmu pronouncing dictionary. <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>.

Aspects of Self-awareness: An Anatomy of Metacreative Systems

Simo Linkola, Anna Kantosalo, Tomi Männistö, Hannu Toivonen

Department of Computer Science

University of Helsinki

{simo.linkola, anna.kantosalo, tomi.mannisto, hannu.toivonen}@helsinki.fi

Abstract

We formulate a model of computational metacreativity. The model consists of various aspects of creative self-awareness that potentially contribute, in various combinations, to the metacreative capabilities of a creative system. Our model is inspired by a psychological view of metacreativity promoting the awareness of one's thoughts during the creative process, and draws from the field of self-adaptive software systems to explicate different viewpoints of metacreativity in creative systems. The model is designed to help in analyzing metacreative capabilities of creative systems, and to guide the development of creative systems in a more autonomous and adaptive direction.

Introduction

Metacreativity, the capability to reflect on one's own creative processes and to adjust them, is an essential part of any creative system that could be claimed to have creative autonomy or intrinsic motivation. For instance, Jennings (2010) argues that autonomous change, the capability of a system to modify its standards by its own decision, is a requirement for creative autonomy. Metacreativity and creative autonomy allow the system to evolve and, eventually, to create artifacts outside the control of the programmer or other agents.

Metacreativity is the subject of various discussions in the field of computational creativity (see, e.g. Buchanan (2001), Colton (2009), Grace and Maher (2015), Jennings (2010) to name a few). Despite a general aspiration towards building systems with greater creative autonomy and a general interest in metacreativity, explicit models of what metacreativity is and how it can be achieved have been scarce. Our goal is to add to the understanding of computational metacreativity by proposing concepts, components and processes useful in characterizing and also building creative systems.

The ability of a computational system to modify itself is known in software architecture research as self-adaptivity (Salehie and Tahvildari 2009; Lewis et al. 2015). We draw from this field of research to derive concepts for metacreative systems.

A key concept for self-adaptivity and metacreativity, both in humans and machines, is *self-awareness*, the ability to be the target of one's own attention. For a system to be

self-aware of some aspect of itself requires that the system explicitly knows of that particular aspect, is able to monitor it and to control itself in relation to it. We formulate these complementary sub-aspects of self-awareness as *(self-)reflection* and *(self-)control*, loosely following concepts used in autonomous computing (Kephart and Chess 2003). The goal is that reflection and control allow a (creative) system to make justified decisions about its own behavior.

Our emphasis on explicit self-awareness implies that the view of metacreativity is “top-down”: metacreative control is located in specific components of the system that have awareness of other parts of the system. The advantage of this modular approach is that it is easier to explicate self-awareness and to discuss how and where a system is self-aware, and the concepts directly suggest possible components and their relations for a creative system. The downside is that the model is not well-suited for systems where creativity is an emergent “bottom-up” property arising from the interaction of multiple (simple) agents or other actors and where there is no explicit self-awareness.

The contributions and structure of this paper are as follows. We start by briefly reviewing the background of metacreativity and self-awareness in computational creativity, psychology and software systems. We then move on to our main contribution: a model for metacreativity, consisting of six different types of creative self-awareness in systems that aim to produce creative artifacts. These aspects of self-awareness can be used to describe existing metacreative systems or to design new ones. After introducing the model, we illustrate how different interesting designs of creative systems can be derived from it. We conclude the paper with a discussion of the model and its relationship to some existing concepts related to metacreativity.

Background

We provide here a brief background on three topics relevant to self-aware metacreative systems: metacreativity as discussed within the computational creativity community, self-awareness in psychology, and self-adaptive software system design.

Metacreativity in Computational Creativity

In the computational creativity literature, the term metacreativity has been used in two kinds of contexts that give it largely opposite meanings. The first meaning refers to the programmer of the system as a metacreator and the created system as a metacreation, to use Veale's (2015) terminology. This view is common especially in the musical metacreation literature (see e.g. Pasquier et al. (2017)). The second meaning of metacreativity refers to a process by which creative systems autonomously evolve their creative capabilities. For example, Ventura (2016) characterises metacreativity as the capability of a system to "change its domain knowledge/summative criteria through learning, interaction, environmental effects". In this paper we talk about metacreative systems, i.e., our use of the term 'metacreation' coincides with the latter meaning.

The few attempts to explicitly model computational metacreativity that we have encountered are based on the idea of transformational creativity. Transformational creativity was defined by Boden (1992) and more formally described in a search-based model by Wiggins (2006). In Wiggins' model, creativity is viewed as search in a space of possibly creative concepts. The search is governed by rules which effectively define a system's search space, the method of traversing it, and a function for evaluating concepts. A system is transformationally creative if it changes any of the rules that govern this search. Grace and Maher (2015) elaborate on Wiggins' model by adding features for modeling an agent's expectations and curiosity during the creative search, and by using this model to evaluate potential rules to direct the transformational search.

Implemented metacreative systems are usually based on the transformational creativity paradigm, by invention of new rules for defining, traversing or evaluating search spaces. For instance, Colton's (2001) HR system, originally designed for mathematical discovery, was extended to discover meta-theories about its own theory formation. Colton argues that these meta-theories could be used to extend HR's creative capabilities by implementing them as new rules for the original program. Similarly, in a more recent attempt Morris, Burton, and Ventura (2012) discuss a culinary system that could achieve meta-level capabilities by transforming its evaluation function over time. This type of computational metacreativity is often implied in works describing transformation of creative search, such as Liapis et al. (2013).

Metacreativity and Self-Awareness in Psychology

Self-awareness is a central concept to our model of metacreativity. Throughout the paper, we use the term as it is defined in psychological literature: self-awareness is the capacity to become the object of one's own attention (Morin 2006). The potential to also change oneself is an important aspect of self-awareness, even if implicit in some discussions. In our model, we make it explicit by talking about self-control in relation to self-reflection.

Bruch (1988) has identified metacreativity as being aware of thoughts and feelings during creative experiences. This

relates metacreativity to self-awareness as defined above. Bruch also relates metacreative acts to Sternberg's (1982) nine element model of problem solving, which takes into account metacomponential executive processes allowing a creator to plan, monitor and evaluate its own creative process.

One form of executive monitoring is constant overseeing of the solution forming process and being aware of new problems – and solutions – which arise during the creative act, e.g., to recognize possibly serendipitous incidents. This kind of monitoring comes close to ideas of solution-focused strategy known as "design thinking", where a designer uses synthesis of previous (sub)solutions to find new solutions, or redefines the problem based on the accumulated information in order to find an acceptable solution (Cross 1982).

A metacreative person needs to be self-aware of his/her own creative processes. Self-awareness, more generally, is an eminent functionality of human cognition. For example, Morin (2006) characterizes the contemporary neurocognitive models of consciousness by the perception of self in time, and by complexity of self-representations. Neisser (1997) formulates five levels of self-awareness which describe the development of higher cognitive functions: (1) ecological self consists of perceptual information, (2) interpersonal self describes raw awareness of social interactions, (3) extended self is able to reflect on itself over time with no explicit focus on mental states, (4) private self can process private information, such as thoughts and feelings and (5) self-concept is made of abstract and symbolic representations of oneself, such as role and identity, in the end encompassing also meta-self-awareness. Neisser's self-awareness levels have served as inspiration for self-adaptive software systems, from which we draw results.

Self-Adaptive and Self-Aware Software Systems

Metacreativity, as the capability of a creative system to change its own creative behavior, is closely related to self-adaptivity in software systems. The challenges of making software systems self-adaptive have been primarily addressed in the domain of software systems (Salehie and Tahvildari 2009; Camara et al. 2016). Self-adaptive software systems are usually closed-loop systems with a feedback loop connecting changes in the operating conditions back to the system (Maes 1987; Salehie and Tahvildari 2009). The changes can originate from the software systems itself or its context, e.g., an operating environment. Self-adaptation requires the system to be able to reflect on the changes, which requires suitable structures of a self-representation of the system (Maes 1987).

Self-adaptation of software is often architecturally based on a general control structure such as the MAPE-K model (Kephart and Chess 2003). The model has two main components: a base system that is being managed and a self-adaptive system taking the control. The self-adaptive system has the capability to reflect on the controlling actions executed over the managed system. The reflection is based on a monitoring component (the M in MAPE-K) that gauges the managed system. The input from the monitor feeds the analyzer (A), which conducts the analyzes for the pur-

poses of planning (P) the adaptive action required. An executor (E) then carries out the action in the managed system. The monitor-analyze-plan-execute loop uses a shared knowledge-base (K) to inform the self-adaptation.

Lewis et al. (2015) elaborate on the basic MAPE-K kind of model using inspiration from the above-mentioned Neisser's (1997) five-level model of consciousness. For conceptualizing self-awareness, they separate and define levels of awareness for self-aware software (Lewis et al. 2015): stimulus-awareness, interaction-awareness, time-awareness, goal-awareness and meta-self-awareness. For the purpose of constructing self-aware software systems, they propose an overall architecture with architectural elements explicitly separating the responsibilities regarding self-awareness. The architecture also distinguishes between private and public forms of self-awareness based on whether a change originates and the awareness relates to something within or outside the system itself, respectively. However, their private awareness concerns the hardware of the systems itself and is not purely awareness about the software, whereas the definitions of self-adaptation focusing more clearly on software may speak of an evaluation on how well the software is accomplishing its task, e.g., in terms of performance (Laddaga 1997).

A Self-Awareness Model for Metacreativity

We next define our self-awareness-based model for metacreativity. We start by defining six possible aspects of creative self-awareness in systems that create artifacts. We then show how different types of systems can be constructed using different combinations of these self-awareness aspects.

Overview of the Model

The model is grounded on the concept of self-awareness as the basis of meaningful self-change in a creative system. We define six aspects of self-awareness for creative systems and outline how these aspects contribute to a creative system's ability to reason about (reflect) and make decisions of (control) its own behavior. These types of self-awareness will be elaborated on later in this section.

Artifact-awareness A creative system that is artifact-aware is able to monitor the artifacts it creates and to adjust what kind of artifacts are generated based on the observed information.

Generator-awareness A creative system that is generator-aware is able to monitor its generator's behavior and adjust it based on the observed information, possibly redesigning parts of the generator.

Goal-awareness A goal-aware system can observe how well it reaches its creative goals, and can modify its behavior and the goals themselves if needed.

Interaction-awareness An interaction-aware creative system knows that some of its actions constitute interactions with other agents or its environment, and can decide how to interact with others in order to influence them or obtain influences.

Time-awareness A time-aware creative system is informed of its behavior in time. It can observe historical development and anticipate likely future phenomena, and modify its behavior based on these observations.

Meta-self-awareness A meta-self-aware system can observe its own self-awareness aspects and influence how they are exercised.

These self-awareness aspects have both implicit and explicit mutual relationships, which are shown in Figure 1. Artifact-awareness and generator-awareness are in a direct hierarchy, as the generator creates artifacts (Figure 1, bottom middle part). Goal-awareness is prominently related to artifacts, but it can also be related to any other element in the system depending on its design. Similarly, interaction-awareness can be in relation to any component of the system, but often deals with exchanging artifacts — or information about them — with external actors. Time-awareness is different from the other aspects, it can only be observed indirectly and it can not be controlled. Meta-self-awareness means awareness about any of the above types of self-awareness; it is also used as an all-encompassing concept meaning the system's ability to reflect and control itself and its own self-awareness aspects.

Self-awareness over an aspect is composed of reflection and control:

Reflection Reflecting on an aspect means tapping into it, monitoring it to gain information about it and possibly processing that information e.g. by generalizing it.

Control Controlling an aspect means adjusting or modifying that aspect.

Both reflection and control are needed for self-awareness: one is meaningless for metacreativity without the other one. They both need a *target*, a component that is monitored and controlled, and another component (or a set of components) that is in charge of reflection and control, dubbed here as the *manager* of the target.

Each of the self-awareness aspects has a variety of possible reflection and control types. They range from simple to complex and, accordingly, we informally talk about *weak* and *strong reflection* and *weak* and *strong control*.

Weak reflection refers to severely limited capability to gain information about the target, e.g. through a black box function. Strong reflection requires analysis of the observations and general attention towards how and what is monitored. A strong form of reflection can be "perceiving": what is observed is externalized from the system (Grace and Maher 2015).

Weak control covers limited adjustments, e.g. parameter changes and other actions with low impact. On the other hand, a system that has strong control over a component may redesign it by changing, adding or removing parts of it.

Aspects of Creative Self-Awareness

Next, we elaborate on each of the self-awareness aspects and argue how they are intertwined with a creative system's capability to gain information of and make decisions about itself.

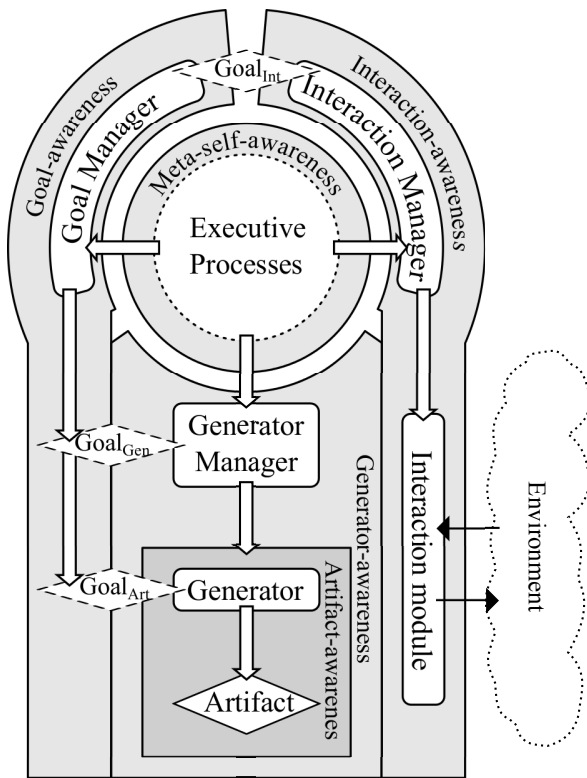


Figure 1: Conceptual drawing of the self-awareness aspects in a creative system and their relations. Reflection and control are indicated by white arrows pointing in the direction of reflection and control. A system can have goals w.r.t. other aspects of awareness and interaction can relate to various other components depending on the system's design. Time-awareness is omitted from the figure, but could be a property of any aspect.

Artifact-awareness A property that distinguishes creativity from mere generation is awareness of the artifacts one generates (Ventura 2016): a system can hardly be called creative if it is not able to assess its own artifacts or adjust what kind of artifacts it generates. An artifact-aware system is not necessarily yet metacreative.

Many creative systems have an evaluation function for artifacts, i.e., they reflect on their products. Some other systems rely on external evaluation; such a system may benefit from building an internal model, e.g. by machine learning, of the external evaluation function. This allows the system then to reflect on artifacts before publishing them, or to modify generation of artifacts to better fit that model.

Consider, for example, a system that generates metaphors, figures of speech where an object (tenor) borrows properties from another object (vehicle). A good metaphor is generally understandable, interesting and tells more about the tenor – or from a different angle – than is usual. If the system is artifact-aware it could, e.g. try to interpret the metaphors it generates using different background knowledge (strong reflection), and modify which object categories (animal, arts,

etc.) the vehicle is drawn from during the generation (weak control).

Generator-awareness The next logical step to increase the creative potential of a system is to make the system aware of its artifact generation process. A system that is generator-aware is able to monitor the artifact generation process and to adjust it if needed. This functionality is called Generator Manager (cf. Figure 1). Generator-awareness gives the system transformational capabilities, allowing it to reach a different space of potentially creative artifacts from what it could reach before.

Returning to the metaphor generator used as an example above, a generator-aware version could monitor the combinations of tenor and vehicle produced (weak reflection), and further analyze if the procedure to select the combination performs poorly (strong reflection). It could then adjust how the combination is selected from candidate sets of tenors and vehicles, e.g. by adjusting weights of matching criteria (weak control).

Goal-awareness Creative systems can have various goals (e.g. maximizing value of artifacts or just exploring a space of artifacts), and the goals can consist of conflicting criteria making their use non-trivial (e.g. how to balance novelty and value if they tend to be negatively correlated?). A goal-aware system knows of its own goal(s) and can use that knowledge to solve some of the issues (e.g. deciding how to strike a useful balance), or it can even modify its own goal(s).

At the most concrete level, the goal is defined as a function over created artifacts ($Goal_{Art}$ in Figure 1). The generator could use this function to reflect on the artifacts it has generated, but mere black-box use of an evaluation function does not count as goal-awareness. For a system to be goal-aware, it also needs to be able to change the evaluation standards.

Depending on the system's design, it can have various other goals potentially associated with other aspects of self-awareness (e.g. $Goal_{Gen}$ in Figure 1). An illustrative example requiring interaction-awareness would be a partial goal ($Goal_{Int}$) on pleasing or provoking the user or other agents in the environment. Goal-awareness therefore is actually an umbrella term for many different possible goals.

Goal-awareness allows the system to further loosen its chains from the developer. The system is then not only able to adjust how the artifacts are generated (if it has artifact and generator-awareness), but it also has the ability to change how it perceives artifacts and what it considers as good or interesting artifacts.

In metaphor generation, the system might monitor the interestingness of tenor and vehicle combinations, along other measures, and its relation to the overall evaluation covering a variety of different measures (strong reflection). This reflection could then have an impact on how interestingness affects the overall evaluation (weak control). For example, in a situation where the current formulation of interestingness

is seen counterproductive, its effect could be diminished in the overall evaluation.

Interaction-awareness Interaction-awareness allows the system to reason and make informed decisions about its environment and how it behaves with respect to the environment. The system knows that it has components that can be used to communicate with outside sources, either directly (e.g. messaging other agents) or indirectly (e.g. leaving pheromones in the environment). Without interaction-awareness, the system does not have an explicit notion of the outside environment, and cannot comprehend that some of its actions constitute communications with others.

A system with strong interaction-awareness can have a primitive *Theory of mind*, meaning, it can model the world outside, including other creative agents and their properties, such as their assumptions about the original system itself. Importantly, it can distinct these from its own properties.

An interaction- and artifact-aware metaphor generation system could model artifact preferences of other agents, and e.g. observe that a certain agent seems to prefer metaphors where the vehicle is drawn from the animal kingdom. It could then use this information to please the agent by communicating to it only with metaphors which have an animal vehicle.

Time-awareness Time-awareness is distinct from the other self-awareness aspects in several mutually related respects. Time cannot be reflected on its own, nor can it be controlled. Time can only be observed indirectly via changes in artifacts, the system's components or its interactions. Accordingly, time-awareness occurs in conjunction with the other types of awareness and, more specifically, their reflections and controls.

For example, a system can be time-aware with respect to its reflection of artifacts. This allows the system to have an understanding of its generation history and to anticipate its future development. Time-awareness with respect to control of artifacts, in turn, allows the system to make plans for what to generate over time (if the anticipated future was not satisfactory). For example, the system could design a strategy of how to approach a perfect instantiation of a certain type of artifact, e.g. an expressionist painting, over time. It could deliberately allocate its resources so that they maximize the strategy's effectiveness, e.g., it can plan to first explore the artifact space more broadly, and to later on focus in sub-spaces with more promising artifacts. Further on, the system is able to monitor and adjust its strategies when needed.

In a metaphor-generation system, time-awareness could be utilized to monitor how novelty or interestingness of the artifacts behaves as a function of time, or how other agents appreciate the generated artifacts during the system's lifespan. This information could then be used to form a plan for what kind of artifacts are communicated to each specific agent and how this can be achieved in a timely manner.

Meta-self-awareness Meta-self-awareness is awareness of one's own self-awareness, so it encompasses all other as-

pects of awareness in a creative system. A meta-self-aware system is able to monitor its own awarenesses and can potentially merge information from different aspects into a unifying view of the system's state.

A metacreative system does not necessary need to have meta-self-awareness, as the system can be metacreative with respect to a single awareness aspect (except artifact-awareness). However, systems without meta-self-awareness have less control over their own behavior.

Consider a meta-self-aware metaphor generation system which has previously generated metaphors with vehicles in the category of animals. Assume that the system observes that it has run out of feasible metaphors with animal vehicles, and that it has previously observed that an agent in the environment prefers metaphors with vehicles from arts. The system could then change its generation of artifacts by fixing the vehicle's category to arts, and change the system's interaction to prefer communication with the art-oriented agent. Further, the system could change its evaluation standards to accommodate the metaphor feedback it receives specifically from this agent.

Connecting Reflection and Control

Self-awareness allows the system to evolve meaningfully during its lifespan. However, it has to have apt connections between reflection and control. We make a distinction between exogenous and endogenous connection types.

An *exogenous connection* is given by an outside source (typically a system's developer), e.g. as a direct mapping between observed and controlled parameters or as a pre-learned model. Exogenous connections do not change during the system's lifespan, and they require that the original source of this connection has a substantial understanding of the problem space at design time, something which is not always feasible – or even desirable. Further, exogenous connections implicitly impose reflection and control types to be well known in advance.

An *endogenous connection* is obtained by actively modeling how reflection and control are related, e.g using machine learning or other adaptive models. Unfortunately, due to the complex or even chaotic nature of many creative applications and phenomena, learning relationships between controls and effects can be hard. Deciding right control procedures for certain reflected elements can require a substantial amount of accumulated information from the problem space, some of which can be obtained by experimentation.

Both exogenous and endogenous connection types are perfectly valid in metacreative systems and can co-exist in the same system. The connections can range from simple, e.g. reflecting and controlling a few parameters in the same component, to sophisticated structures connecting reflected elements from various self-awareness aspects to the controls of others. The most eminent connections operate also on meta-self-awareness and execute complex reasoning about how certain reflection should be effected in control.

Example Configurations

We next outline some example configurations of self-awareness aspects and give name suggestions for them.

Creative We call a system creative (as opposed to merely generative) if it is aware of its own artifacts and uses strong reflection and control over them in an exogenously or endogenously connected manner. The behavior of such a system changes over time as it controls new artifacts based on what it already has generated. This does not necessarily imply time-awareness, however.

Self-transforming A self-transforming system modifies its generator in a way that allows it to reach previously unattainable areas of a conceptual space. That is to say, the system is generator- and artifact-aware, and can utilize an exogenous or endogenous connection between them to come up with new generators enhancing the system's capability to explore the creative space.

Self-guiding A system is said to be self-guiding, if it is able to make justified long-term plans and change its way of generating the artifacts if the plan is failing. This requires the system to have generator-awareness and time-awareness. A self-guiding system looks at its own generation process history and estimates how it will be doing in the future if the generator stays unchanged. If the system predicts that the current way of generating artifacts is going to be inadequate in the future, it modifies its generator and starts to make new estimations about its generator's future competence.

A self-guiding metaphor generation system may plan to try to generate as many valuable metaphors as possible with a generator that specifies a set of properties the vehicle must have. If the system anticipates that it is not able to generate any more valuable metaphors with such constraint in its generator, it may change the constraints, e.g. by specifying another set of properties.

Autonomously creative A system is autonomously creative if it changes its goal(s) (Jennings 2010) based on the information it gains from reflecting over the artifacts. Such a system is both artifact-aware and goal-aware. The connection between the artifact reflection and the goal changes can be either exogenous or endogenous. However, an endogenously connected system could be seen to have greater internal motivation.

The metaphor generation system could learn via reflection that vehicles with some properties never score highly in the overall evaluation with current constraints (reflecting on artifacts). It could then directly modify its goal and generator to avoid such vehicles (control of goals, control of generator).

Collaborative A collaborative system interacts with other agents in the environment to advance a common creative goal. It requires interaction-awareness and goal-awareness from the system, as the system has to be able to control its communication and adjust its own goals based on the exchanged messages.

A set of collaborative metaphor generation systems could, for example, divide the metaphor search space so that each agent operates on a distinct subspace. This requires

communication-aware coordination of how to set the individual goals of systems.

Self-driven A self-driven system is both self-guiding and autonomously creative. The system makes long-term plans to acquire feasible strategies for future behavior, and changes its goals based on its behavior (e.g. by reflecting on the artifacts or the current generator) and the plan it currently adheres to. To rigorously adapt to new plans and goals, a self-driven system needs to have endogenous connections between reflection and control, requiring meta-self-awareness to connect different self-awareness aspects.

A self-driven metaphor generation system could form a similar plan as in the self-guiding example: fixing a set of properties a vehicle must have and trying to generate as many valuable metaphors as it can. Then, when it anticipates that it cannot generate any more metaphors with current constraints, it may form a new plan with a new set of properties the vehicle must have. At this point, it also can directly modify its goal and generator to avoid vehicles with the former set of properties.

All the above configurations except for "Creative" are metacreative in a meaningful manner. For example, a self-guiding system could be said to have a primitive form of intent as it operates proactively and carries out previously made plans, and an autonomously creative system satisfies the requirements for creative autonomy by Jennings (2010).

However, there still is a gap between these configurations and a system which is in complete control of its own development as a creator, as even a self-driven system is not aware of its own self-awareness and thus cannot change how it is aware of artifacts, the generator, or its own goals. The next level of metacreativity thus involves meta-self-awareness, an executive process that manages other self-awareness aspects. It regulates their operation in order to best fulfill the system's current goals, and allowing it to temporarily – and intentionally – concentrate on specific creative subspaces. We return to this topic in Discussion and Conclusions.

Discussion and Conclusions

We have presented a model for metacreative systems, heavily based on the concept of self-awareness characterized by self-reflection and self-control as its components. In our model, a metacreative process evolves through apt connections between reflection and control, the connections being either exogenous (given, static) or endogenous (achieved or learned by the system). The proposed model is modular in the sense that the six self-awareness aspects introduced can be combined in numerous ways to reach various configurations of metacreative systems.

The proposed model does not cover all possible types of metacreativity, and even within the covered aspects there is a large variety of nuances that could be characterized in more detail. However, we believe that the model contains a representative and diverse set of high-level aspects in which creative systems are potentially metacreative. We believe that the concepts introduced here help computational creativity

researchers better analyze and describe how their systems are metacreative.

Designing metacreative systems The model can also be used as a software architecture for metacreative systems. In Figure 1, all boxes with rounded corners are possible components of a metacreative system. For instance, the Generator Manager stands for the functionality that observes and controls the generator. Implementing it as an explicit component (or a set of them) allows, in turn, a meta-self-awareness component to observe and control it.

For the design of metacreative systems, two main lessons can be taken from software architecture design: separation of concerns and means for increasing flexibility.

Separation of concerns is of importance for explicating the responsibilities and functionality related to different aspects of metacreativity and, in particular, clearly defining the levels of metacreativity in software.

On the other hand, to implement metacreativity, the target elements to be modified, e.g., the Generator Manager, need to provide the means for modifiability. A principle mechanism towards modifiability is modularization that allows access to the mechanisms controlling the element. The actual modification mechanisms may include manipulation of the code at runtime, parametrization of computation, dynamic (re)configuration of the functionality, e.g., by means of plugin components or more sophisticated means of model-based configuration.

MAPE-K Drawing from research in self-adaptive and autonomous systems, our model is inspired by MAPE-K (Kephart and Chess 2003). Reflection in our model can be seen as a mix of monitoring (M) and analysis (A), control as execution (E), and connections between reflection and control as a form of planning (P). However, we talk about reflection and control since they simplify the model, and since they also better support describing situations where reflection and control span multiple self-awareness aspects, possibly including meta-self-awareness capabilities.

Aspects of self-awareness Two of the self-awareness aspects, artifact-awareness and generator-awareness, are specific to and especially interesting for creative systems. A similar separation of artifact and process levels has been previously brought up in computational creativity literature, e.g. by Ritchie (2006), O'Donoghue et al. (2014) and Wiggins (2006). The other four aspects are more general in nature and are inspired by self-awareness levels in self-adaptive systems (Lewis et al. 2015). However, they are formulated here by taking into account specific points of view related to creativity, and have been augmented with a control requirement.

Relation to other models in computational creativity Even though specific self-awareness aspects are not usually presented in computational creativity models, our model can be used in conjunction with existing models of computational creativity. For example, Colton, Charnley, and Pease (2011) argue in their FACE model that creative systems should be able to explain and justify their creations or processes to their audience by generating framing information. The self-aware aspects of our model offer concrete topics and processes for generating such framing informa-

tion. For instance, an artifact-aware system can explain its artifacts, a generator-aware can further reflect on how it obtained them, a goal-aware system can talk about its goals and can relate the results and process to the goals.

Wiggins (2006) models creativity as search and metacreativity as transformational creativity. Wiggins' meta-level could be seen parallel to our generator-awareness, focused on changing the generation process based on information about the process itself. Our model gives several additional concepts that help analyze and describe metacreativity beyond artifact or process levels. Further on, with our model we can directly point out meta-level elements which allow a system to escape unwanted creative behaviors described by Wiggins (2006). For example, a self-aware system can get over *generative uninspiration* — the system's inability to find valued artifacts or concepts — by intentionally interacting with others to obtain new seed artifacts, by modifying its own goals, or by changing how the generator traverses the creative space.

Serendipity A creative system arrives at a *serendipitous incident* when the system realizes that it has unintentionally done something valuable. With our model, we can describe systems which may take advantage of these situations. For example, a self-driven system is able to recognize them: it is able to assess the value of the incident, and assess if following it pays off better in the long run than the current plan. That is, it does not follow the serendipitous incident blindly, but can form a new action plan from it.

Role of meta-self-awareness In our model, meta-self-awareness is the birthplace for many high level creative phenomena. For example, specific curiosity, characterized by Grace and Maher (2015) as a driving force of intentional transformational behavior, would need elaborated meta-self-awareness. Without meta-self-awareness, a system cannot have a unified view of its state and where it should direct its attention. This may cause the system to behave erratically as the fragmentary curiosity of different system components is rendered contentious.

To give a system the ability to fully be in charge of its own development, the connections in the meta-level executive processes have to be (partially) endogenous. A particularly suitable class of machine learning models for these connections are intrinsic motivation models (see, e.g. Oudeyer and Kaplan (2007)). If appropriately applied, they impose on a system an evolving attention towards its own operation and goals. As such, they inherently enable experimentation and are a good fit to the open-endedness of many creative tasks.

Some intrinsic motivation models may give a system a conation not only to try out new reflection and control connections but also allow it to build up competence towards temporarily fixed goals. For example, these models can be applied to naturally inhibit some sensory stimuli (e.g. specific measurements from artifacts) for a while and concentrate on others (e.g. communication responses of specific agents) and continuously re-evaluate and modify these configurations during the system's lifespan. In this sense, they are promising candidates to offer means for a meta-self-aware creative system to, e.g., exhibit diversive and specific curiosity as discussed by Grace and Maher (2015).

To conclude, we have described six different aspects of self-awareness that creative systems possibly exhibit. As a conceptual tool, these aspects can be used to describe, analyze and compare creative systems. These aspects also directly suggest potential building blocks for metacreative behavior in creative systems.

The next obvious step is to use this model to analyze and describe existing metacreative systems, to test its value as a descriptive and analytical tool. An interesting topic for future work is also designing example systems to concretely illustrate and test some of the self-awareness aspects presented in this paper. Both the process of creating such a system as well as the end result will be important in showing the value of the model as an architecture for metacreative software.

Acknowledgements

This work has been supported by the Academy of Finland under grant 276897 (CLiC).

References

- Boden, M. 1992. *The Creative Mind*. London: Abacus.
- Bruch, C. B. 1988. Metacreativity: Awareness of thoughts and feelings during creative experiences. *The Journal of Creative Behavior* 22(2):112–122.
- Buchanan, B. G. 2001. Creativity at the metalevel: AAAI-2000 presidential address. *AI Magazine* 22(3):13–28.
- Camara, J.; Lopes, A.; Garlan, D.; and Schmerl, B. 2016. Adaptation impact and environment models for architecture-based self-adaptive systems. *Science of Computer Programming* 127:50–75.
- Colton, S.; Charnley, J.; and Pease, A. 2011. Computational creativity theory: The FACE and IDEA descriptive models. In *Proceedings of the Second International Conference on Computational Creativity*, 90–95.
- Colton, S. 2001. Experiments in meta-theory formation. In *Proceedings of the AISB01 Symposium on AI and Creativity in Arts and Science*.
- Colton, S. 2009. Seven catchy phrases for computational creativity research. In *Computational Creativity: An Interdisciplinary Approach*, number 09291 in Dagstuhl Seminar Proceedings.
- Cross, N. 1982. Designerly ways of knowing. *Design Studies* 3(4):221 – 227.
- Grace, K., and Maher, M. L. 2015. Specific curiosity as a cause and consequence of transformational creativity. In *Proceedings of the Sixth International Conference on Computational Creativity*, 260–267.
- Jennings, K. E. 2010. Developing creativity: Artificial barriers in artificial intelligence. *Minds and Machines* 20(4):489–501.
- Kephart, J. O., and Chess, D. M. 2003. The vision of automatic computing. *Computer* 36(1):41–50.
- Laddaga, R. 1997. DARPA broad agency announcement on self-adaptive software (BAA-98-12).
- Lewis, P. R.; Chandra, A.; Faniyi, F.; Glette, K.; Chen, T.; Bahsoon, R.; Torresen, J.; and Yao, X. 2015. Architectural aspects of self-aware and self-expressive computing systems: From psychology to engineering. *Computer* 48(8):62–70.
- Liapis, A.; Martínez, H. P.; Togelius, J.; and Yannakakis, G. N. 2013. Transforming exploratory creativity with DeLeNox. In *Proceedings of the Fourth International Conference on Computational Creativity*, 56–63.
- Maes, P. 1987. Concepts and experiments in computational reflection. In *Conference Proceedings on Object-oriented Programming Systems, Languages and Applications*, OOPSLA '87, 147–155.
- Morin, A. 2006. Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. *Consciousness and Cognition* 15(2):358 – 371.
- Morris, R.; Burton, S.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *Proceedings of the Third International Conference on Computational Creativity*, 119–125.
- Neisser, U. 1997. The roots of self-knowledge: Perceiving self, it, and thou. *Annals of the New York Academy of Sciences* 818(1):19–33.
- O'Donoghue, D. P.; Power, J.; O'Briain, S.; Dong, F.; Mooney, A.; Hurley, D.; Abgaz, Y.; and Markham, C. 2014. Can a computationally creative system create itself? Creative artefacts and creative processes. In *Proceedings of the Fifth International Conference on Computational Creativity*, 146–154.
- Oudeyer, P.-Y., and Kaplan, F. 2007. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurobotics* 1(6):1–14.
- Pasquier, P.; Eigenfeldt, A.; Bown, O.; and Dubnov, S. 2017. An introduction to musical metacreation. *Computers in Entertainment* 14(2):2:1–2:14.
- Ritchie, G. 2006. The transformational creativity hypothesis. *New Generation Computing* 24(3):241–266.
- Salehie, M., and Tahvildari, L. 2009. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems* 4(2):14:1–14:42.
- Sternberg, R. J. 1982. Teaching scientific thinking to gifted children. *Roeper Review* 4(4):4–6.
- Veale, T. 2015. Computational approaches to language and creativity: Creativity ex machina. In Jones, R., ed., *Routledge Handbook of Language and Creativity*. Routledge. 353–366.
- Ventura, D. 2016. Mere generation: Essential barometer or dated concept? In *Proceedings of the Seventh International Conference on Computational Creativity*, 17–24.
- Wiggins, G. A. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19(7):449–458.

Application Domains Considered in Computational Creativity

Róisín Loughran and Michael O’Neill

Natural Computing Research and Applications Group (NCRA)
University College Dublin
Ireland
roisin.loughran@ucd.ie

Abstract

We present a review of papers presented at IJWCC and ICCC, specifically considering what applications these papers are engaged with, either directly in generative systems or indirectly in evaluation or framework proposals. The primary focus of this work was to ascertain if there are any trends in the applications considered over the years, any topics that are becoming more dominant or any that have been neglected. Our initial classification among 16 specific categories indicated that Music was the most popular application domain; when we reconsidered seven broader categories we determined that papers involving variations of language processing were most popular. We considered the trend among application domains over the past 12 years and noted that contrary to early discussions on creativity, problems based on logic, science or mathematics do not appear often. We consider the implications of this research as to what information it may convey both to the computational creativity community and to a general computer science audience.

Introduction

A Computationally Creative system is defined as one that can be shown to exhibit behaviour deemed to be creative (Colton, Wiggins, and others 2012). A concise, generalised and context-free meaning of the term *creative* has yet to be defined, however. As such, many scientific studies in the field of Computational Creativity (CC) develop and describe systems that exhibit creativity in a specific application domain. Discussion and evaluation of such systems is then dependent on their ability to function within the given domain. Although some studies within the CC field consider creativity in a more generalised sense with no domain in mind, the majority of papers — even those that are not specifically describing a system designed to produce a single artefact — discuss the merits of the work undertaken in relation to one or more specified applications. This paper takes a quantitative examination of the application domains considered in CC research, specifically from those papers published by the CC community at the main annual events from 2004 to 2016. For this study we consider each paper individually and make a subjective categorisation, rather than using any autonomous, lexical classification techniques.

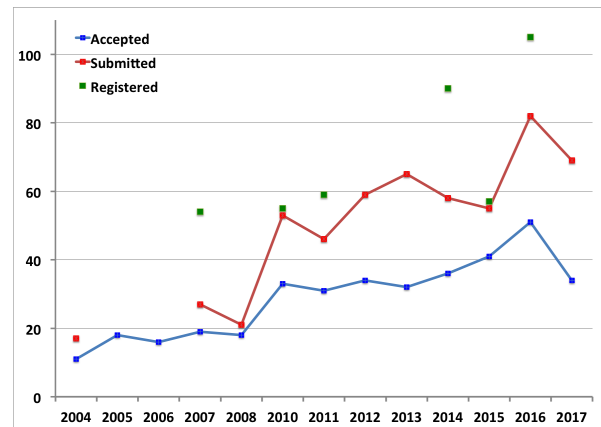


Figure 1: Number of registrations, papers submitted and papers accepted at IJWCC and ICCC from 2004-2017

CC is a young but expanding field that has been gaining momentum over the last decade. After the International Joint Workshops in Computational Creativity (IJWCC) held as part of larger conferences in 2006 and 2007, the first stand alone workshop was held in 2008. This was further developed into the first International Computational Creativity Conference (ICCC) in 2010 which grew steadily in the following years; ICCC16 had 51 published papers with over 100 registered attendees for the first time. The growth of the event in terms of number of registrations (where available) and number of papers submitted and accepted is illustrated in Figure 1¹. This indicates a general increase in participation and interest in the event over the years. The ICCC conference series is the only scientific conference devoted entirely to all aspects of CC. As such, the proceedings from these conferences offer a comprehensive insight into the work that has been undertaken by leaders of this field. The work submitted, reviewed and presented at these events shapes the field of CC and the direction in which it is going.

This paper considers application domains investigated in this field, not to find the best domain or negate the value of any specific domain but rather to establish if there is a trend

¹Numbers kindly supplied through personal correspondence from members of the Association for Computational Creativity.

in the application domains under consideration by the community, and what — if any — implications this may have for the field as it progresses. CC is still a relatively young field, and yet it encompasses an extremely broad range of topics. As such, it is important to regularly take stock and review any direction it may be taking. The following section offers further discussion on our motivation for this study and what we hope to achieve in undertaking it. The remainder of the paper describes the method by which we categorised the reviewed papers and discusses the results obtained and what conclusions we may draw from them.

Motivation

We conducted the proposed study to review the application domains within which contemporary research on CC is being considered. The term *creative* is one which is colloquially understood and yet inherently difficult to define in a generalised context. In discussions on creativity, either human or machine, there is a natural tendency to use the generation of examples of creative artefacts to demonstrate creative behaviour. Such examples will belong to a specific application domain e.g art, music or literature, but while creativity may be exhibited as such, creativity in general is not specific to any given application, artefact or domain. In humans, achievements in such domains are generally attributed to creative ability — people that are artistic, musical or poetic are often described as creative. Such ideas lend to the notion that any creativity requires special ability, that being creative is only exhibited as an aesthetic talent to be honed and nurtured by the few that display it, rather than an innate ability possessed by us all. Creative behaviour is not limited to remarkable achievements in aesthetic domains however. Personal or P-creativity (as opposed to Historical or H-creativity) is a personal creative ability all people possess, displayed in the generation of an idea that may already exist in history but is new to the individual (Boden 1998). Most people use P-creativity regularly, any time they solve a problem or have a new idea. In studying creativity in a scientific manner, it is this P-creativity that is of most interest, regardless of the domain in which it is demonstrated.

CC is still a young inter-disciplinary field, but as can be seen from an increasing number of publications, it is a field that is growing. Within any growing field, new studies build on research previously undertaken, and early studies are often considered pioneering work in the area. Preliminary studies are continued, developed and used as the basis for more mature studies. As the field develops, ideally we would wish for a balanced approach to the domains being used for applications. If we assume that the presence of creativity is not dependent on the application domain, we must consider that any continued focus towards one domain over another may introduce a bias within the field in general. Furthermore, to encourage development of the field and to attract new researchers and talent to the area we may wish to ensure that new applications are continuously being explored and that the domains considered do not become stagnant.

It is important to note that papers are discussed here *only* in terms of which application they use — even if this application is not the primary focus of the work. Some papers

are application-focussed while others merely use the application in their discussion of a broader creativity issue. We wish to help avoid the automatic use of applications without any more justification than they have been ‘used before’ and thus gather inertia to remain dominant within the field. Such problems have been noted to emerge in evaluation within other applied computing areas, for instance, using ill-suited benchmarks that have been noted to persist in the field of genetic programming (McDermott et al. 2012).

There have been a number of previous studies concerned with the direction and development of the field of CC. A review of the history (and predicted future) of the field was offered in 2009 in the interim between the IJWCC and the start of the ICCC conference (Cardoso, Veale, and Wiggins 2009). This paper considers creativity in general and reviews a number of approaches that have been undertaken in CC along with challenges and progress in the field. It concludes with an optimistic outlook for development within field — one that has been seen to come to fruition with the increasing success of ICCC in the intervening years. Many other survey style papers focus on one aspect of CC, such as problems in evaluation. The difficulty in defining creativity naturally leads to a resultant difficulty in evaluating a creative system. This has led to a number of authors doing self evaluation, minimal evaluation or no evaluations at all on their systems. The lack of evaluation in CC systems has been noted throughout the development of the field (Boden 1998; Cardoso, Veale, and Wiggins 2009; Jordanous 2011). Such studies highlight the need for a clear definition of what can be considered creative.

A method of semi-automated domain conceptualisation on papers from the last six years of ICCC was proposed in (Pollak et al. 2016). The current paper differs in that it offers no automation in the categorisation of applications. We consider each paper individually and determine the domain from the discussion given by the author rather than any information extraction or analysis of the syntax in the papers. The purpose of this work is to consider the domains in which the academic discussion on CC has been undertaken in recent years, specifically by looking at publications at ICCC, and to determine if there are any trends worth noting and what such trends might mean for the overall CC community. This review would be of use to CC researchers concerned with the progress of the field, but also to those new to the area that wish to know what problems have been addressed so far by established researchers. In addition to this methodical review, we wish to reflect on how a focus on domain-based results may be influencing current and future methods of evaluating creativity.

Analysis

In total, 353 papers from 12 years were considered. The number of papers submitted and accepted each year is shown in Figure 1. Table 1 gives the names of sessions that were incorporated in each event. While many of the session names share similarities across years, it is clear there are no ‘standard’ categories that all papers must fall into. For the purpose of this study, we considered each paper individually and assigned it to a specific category.

Table 1: Overview of organisation of papers in each year

Event	Sessions
IJWCC04 Part of ECCBR04	unspecified
IJWCC05 Part of IJCAI04	Mathematical and analogical creativity, Theoretical issues in computational creativity (x2), Creativity in the literary domain, Creativity in the music domain, Creativity in other human activities
IJWCC06 Part of ECAI04	Visual creativity, Musical creativity (x2), Frameworks, Linguistic creativity (x2)
IJWCC07	Creativity in narrative, Analogy and language, Musical creativity, Applied creative systems, Frameworks for creativity
IJWCC08	Theory of creativity, Techniques to get creativity, Storytelling, Music, Platforms and experimental frameworks
ICCC10	Music: patterns and harmony, Visual art, Analogy and metaphor, Stories, Social aspects, Foundations, Music: creation/generation, Creativity support: tools
ICCC11	The Applied, The Social, The Narrative, The Cybernetic, The Foundational, The Helpful, The Cognitive, The Exploratory
ICCC12	Conceptual blending, Analogy, Search, Reflections, Generative systems, Evaluation (x2), Computers being creative, Cognition and computation, Creativity and language
ICCC13	Metaphor in computational creativity, Creativity via computational evolution, Creative processes, Music, Visual art, Computational processes for creativity, Evaluating computational creativity, Poetry, Narrative, Collective and social creativity, Embodied creativity
ICCC14	Co-creation, Visual arts, Videogames, Poetry, Music, Evaluation, Evaluation/Data, Language/Narrative (x2), High level issues
ICCC15	Creative autonomy, Evaluation in the arts, Creative mechanisms, Language, Evaluation of creativity, Musical interaction, Conceptual blending, Visual arts, Games music and cocktails, Creativity support, Imagination and curiosity, Co-creativity, Language
ICCC16	Search, Evaluation, Interaction, Models of creativity, Visual arts, Narratives, Language, Generating structure, Beyond the fence, Blending, Software platforms

Table 2: Description of the 16 initial categories and the higher-level grouping each category was assigned to

Category Name	Description	Higher-level Grouping
Story	story-telling, plot development, character development	NLP
Language	general language syntax, lexicology, translation	NLP
Analogy	analogy and metaphor (text-based)	NLP
Literature	poetry, haiku, sonnet generation or analysis	NLP
Humour	language systems based on understanding or generating humour	NLP
Design	design implementation, description or augmentation	Other
Coding	programming and generating coding solutions	Other
Games	generating, augmenting or playing computer games	Other
Other	any specifically named system not in one of the named categories	Other
Sound	sound generation and analysis, sound effects	Music
Music	music generation, analysis or composition	Music
Maths (and Science)	mathematical formulae, scientific problems, numerical problems, theorems	Logic
Logic	logical problems, general problem-solving	Logic
Image	image generation, analysis or composition	Image
Concept	general high-level concepts (not text-based)	Concept
None	papers that do not discuss any application	None

Categorisation of papers

Categorisation was conducted subjectively by the authors through a review of each paper. This categorisation was performed personally rather than using autonomous, lexical classification to ensure we encapsulated the intended application domain of the author. Using an autonomous, statistical analysis of the papers would likely produce different results, but it was author intent that we found to be more interesting, particularly in view of what this may say about the direction of the the field, as discussed later in the paper.

Many CC papers are based on generative systems. Such systems are trivial to categorise from an application domain perspective; a system that generates paintings is clearly in the domain of visual art or images. A large number of systems are not so specific however. The Call for Papers for these events have always supported the submission of pa-

pers on general creativity, high level concepts or position papers that discuss developments within the field. Such studies often do not specify any application domain. For the purpose of this study we assign these papers to the category of ‘None’. Likewise, some papers mention multiple application domains. Papers based on evaluation of creativity may present results in a number of different domains. In these cases the paper is placed in multiple categories. For example, (Kantosalo et al. 2014) is considered to be in the categories Humour, Choreography, and Design. Of course, this is only the categorisation for the proposed work; the primary focus in this paper is in investigating human-computer co-creation. In this way, the categorisations proposed here do not necessarily correlate with the session organisation as detailed in Table 1. We are purely considering papers from application domains explicitly stated by the authors.

From inspecting each paper in the catalogue we identi-

fied 16 initial individual categories: Logic, Story, Language, Analogy, Sound, Design, Maths, Image, Music, Literature, Concept, Humour, Coding, Games, Other and None. These were chosen as the main topics described explicitly by authors in numerous works. A brief explanation of each category and which papers were included in each is given in Table 2. While most of these are self explanatory as practical applications, one notable exception is the category of ‘Concept’. This category described papers that were not purely positional — they described experiments and offered results — but focussing on higher-level concepts or ideas, rather than a specific physical object as the artefact associated with their work. A number of papers focussed on conceptual blending, such as (Martins et al. 2016), were best categorised as ‘Concept’.

As expected some papers were easily categorised, but many were found to be more difficult to attribute to one individual category. Only papers that explicitly stated more than one application were given multiple categorisations; papers whose application domain was ill-defined or appeared to span multiple domains in one study were subjected to a judgment on our part and assigned to one category. This category was always chosen as that which appeared to be in focus from the authors perspective in discussing their work, rather than making a judgement based on the title, abstract or which session it was included in. For example, (Ventura 2008) could be considered a theoretical or concept-based paper, yet it discusses hypothetical images. Although no images were created by this system, it has been categorised as an ‘Image’ paper as this is the way the paper has been discussed. Other papers arguably could be categorised as either concept or analogy, or possibly story or analogy. Again in such circumstances we categorised in favour of the discussion presented in the individual papers. Certain papers raised severe difficulties in categorisation. (Johnson 2012) mentions nearly all aesthetic fields — music, art etc. yet the overall discussion is mostly concerned with the definition of CC. Arguably, such a paper could be considered to have almost all applications or none. In this case, we have categorised it as ‘None’. A small number of papers required such a subjective categorisation. For complete transparency, a full list of each paper and which category we attributed it to is available in the accompanying appendix: <http://tinyurl.com/lg2aqq4>.

Throughout this discussion, no distinction has been made between long and short papers or those that were presented orally or as posters. However, Demonstrations and Show and Tell sessions were not included in this paper.

Category reduction As described above, these 16 categories were chosen according to the application domain specified by the authors. Many of these initial categories share similar properties and could be amalgamated into broader groupings; there is no one ideal number of categories in such a study. For an alternative level of analysis we reduced the number of categories by grouping together those that could be considered similar. We reduced the 16 sub-categories into the 7 higher-level categories as detailed in the third column of Table 2. As evident from this table many of

the subcategories can be re-categorised as Natural Language Processing (NLP). This covers any application that directly involves text analysis and understanding. Notably we did not consider ‘Concept’ to be part of this grouping as those papers categorised as Concept were not text-based but considered the notion of a concept as a higher level or abstract idea. This greatly reduced number of application categories enables a clearer analysis of the results reported in the following section.

‘Other’ expansion Conversely, the single category of ‘Other’ clearly can refer to a large number of subcategories of applications. Any application that is specifically named but does not belong to one of the categories defined above is considered to belong to Other. It is arguably possible to again consider some of these as sub-categories of more generalised applications described above. For instance, Archaeology is given as the application in one early paper (Cos et al. 2007) and while on inspection this does amount to image analysis, the authors have framed and written the paper from the perspective of Archaeology. Again in cases such as this, where a novel application has been explicitly mentioned by the author, we have chosen this as the given application domain. While presently, these are all categorised as ‘Other’ we consider individual applications and the increase in the use of specific topics in recent years in the results below.

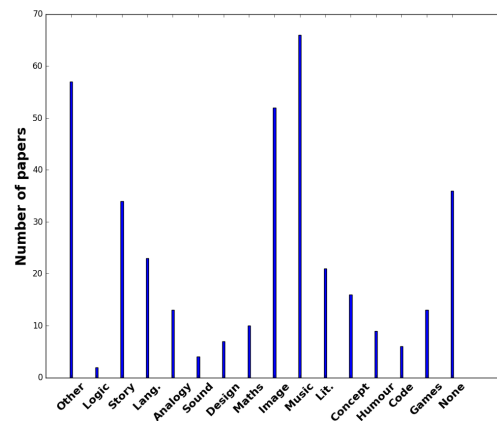


Figure 2: Number of papers in each category, 2004-2016

Results

An overview of the total numbers of papers submitted in each of the original 16 categories summed over all 12 years is shown in Figure 2. This indicates that Music is the most popular single category across all years, followed by Other and Image. The number of papers categorised in to the reduced number of categories, as specified in Table 2 is displayed in Figure 3. It is clear from this figure that when considered with this categorisation, papers based on NLP are actually more popular over the years. This may be unsurprising as the topic of NLP encompasses numerous smaller

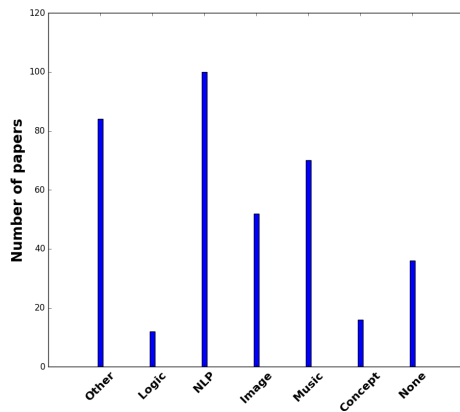


Figure 3: Number of papers in each (reduced) category, 2004-2016

yet always popular categories such as Storytelling, Literature (including poetry) and Humour. It is not just papers based on Language whose main application is considered NLP, but those whose application inherently require a semantic understanding such as those in Analogy or Humour.

The percentage of papers in a given category in each year is shown in Figure 4. This shows the trend in applications considered across all years. One point to note from this graph is that the ‘Other’ category has become more prevalent in recent years; this category has had the highest, or second highest, percentage of papers in every year since 2013. We can see from Table 1 that there have been more papers accepted since 2014; as the field has grown and more papers are being written, there are more papers considering new applications. This growing diversity can only be beneficial to the field of CC in general as it indicates new areas of interest, new ideas and new problems being considered.

One surprising result evident from Figures 2 through 4 is the lack of papers based on Logic, Mathematical or Scientific problems. Over all years, studies based on scientific problems or applications have not been popular among CC papers. This is quite surprising when we consider that early discussions on Creativity were often illustrated with scientific, logical or mathematical problems. Much discussion on creativity by Boden is on the scientific and mathematical works of Poincaré, Kekulé and Einstein (Boden 2004) — a point reiterated in the discussion on the development of CC (Cardoso, Veale, and Wiggins 2009). Despite this, papers written by the CC community have focussed on the more traditionally creative or aesthetic applications such as music, art and literature.

Other applications

We have noted that the category of ‘Other’ has become increasingly popular in recent years. This covers the generalisation of topics that have only appeared once such as Cocktail preparation (Pagnutti and Whitehead 2015), to others such as Choreography that may have originally been

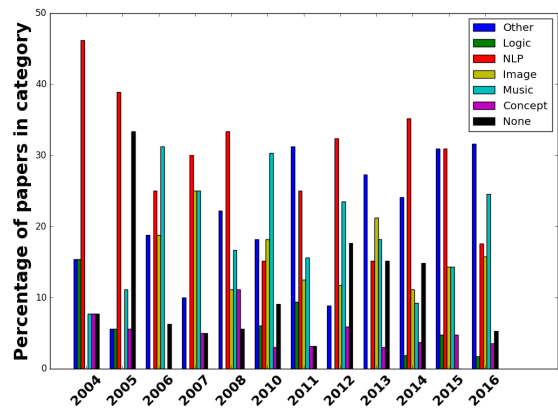


Figure 4: Percentage of papers in each category, 2004-2016

considered obscure but have now gathered a following. Choreography in particular has been the main application for six individual papers — four of which were in 2016. We have not explicitly defined a threshold number which must be reached for a topic to become a category, but if choreography remains this popular in the next few years it would rightfully be considered a category of its own. Often it does not take many publications on a topic for it to be considered typical within the field. For instance there were just three individual papers published on recipe creation (Butnariu and Veale 2006; Morris et al. 2012; Shao, Murali, and Sheopuri 2014) before this was chosen as a topic for discussion within a framework paper (Grace and Maher 2015). Many of these Other applications draw from variations or combinations of the categories defined above, but place them in a specific context. For example Internet Memes (Costa, Oliveira, and Pinto 2015) are a specific combination of Humour and Image, or Computer Icons (Confalonieri et al. 2015) are a combination of Image and Design. The most ambitious singular application undertaken to date is surely the Beyond the Fence musical (Colton et al. 2016). The creation and production of this musical was the result of a collaboration from researchers with experience in a wide range of application domains. Large scale projects such as this that may be considered a unique (or ‘Other’) application will undoubtedly always result from a combination of other established application domains.

Papers were considered to be within the Other category when the authors stated the application domain explicitly. A complete list of Other topics that have occurred at least once in this literature includes: role-playing games, map generation (2), puzzles, evolutionary robotics (3), archaeology, cinematography, improvisational theatre, furniture arrangement, chat communication, advertising (2), exploratory gene analytics, play, web-comics, choreography (2), identity structures (2), visual narratives, recipes (3??), subvertising, dementia care, flowcharts (2), modelling, animation (2), interior design, agents in 3D environments, travel, fashion, computer icons (2), 3D vases, cocktails,

kinematics, authoring, scientific discovery, internet meme (2), maze navigation, internet movie database, 3D objects from 2D objects, Beyond the Fence (2)².

Session organisation

The names of the individual sessions taken from Programs and Proceedings at each event is given in Table 1. It is interesting to note that the naming of the organised paper sessions differ from year to year. In some years, the session names have been very application focussed whereas in others they were not. Naturally this is at the discretion of the organisers but this variety in session naming indicates a fluidity within the development and progression of focus in the field. Interestingly, the number of papers related to a given application did not necessarily directly influence the choice of session names. For example, there is a session name focussed on music in each year except 2011, 2012 and 2016 (there are in fact two sessions on music in 2010). However, these years did not lack in papers focussed on musical applications. From Figure 4 we can see that in fact in 2012 and 2016, Music was the second highest application domain represented in accepted papers.

Discussion

We have presented a study focussed on application domains within CC research, but we are not attempting to establish one ‘most creative’ domain. Contrary to colloquial sentiment there is no one domain that is more creative than another; it would be very difficult to determine if an autonomously generated recipe for curry displayed more or less creativity than an autonomously generated piano melody. Absurd as this comparison may seem, if we measure the creativity exhibited by a system purely on the output produced, this type of comparison would become inevitable. Such situations can only be avoided through domain-independent evaluation of the system. Merit should be assigned to progress in the creativity exhibited by a system, rather than to superficial adjustments that merely change the output of a system in a given domain. Systems that produce artefacts requiring more domain knowledge or more complex representation can appear to be more impressive than those that create in simpler domains. An increase in complexity does not necessarily imply an increase in creativity however. It is vitally important when evaluating creative systems that it is the system being evaluated — the processes it undertakes to create, given the domain knowledge that has been presented to it. For evaluation to be domain independent, it must take into consideration all domain knowledge learned by or available to the system, and somehow measure the leap that the system made from this knowledge to what it was able to produce. Given the current state of a system, the representation it uses and the training data, grammars or other a priori information it has access to — what intuitive leap does it make in creating its output? In asking this we may first wish to consider if an autonomous system can actually make an ‘intuitive’ leap, or if it can only

²A graphical display of the spread of these topics was not possible, but those that had multiple instances are shown in parentheses.

be considered intuitive once a person acknowledges it to be so? Furthering this we may have to ask: Can a computer be creative if there is no-one there to call it creative?

The current definition of a CC system is one which ‘exhibits behaviour deemed to be creative...’ thus it is the *behaviour of a system* that needs to be evaluated; the application domain is merely the setting for the experiment. There is a circular, self-referential issue in that CC is defined in terms of creative behaviour, which is often displayed in the creation of artefacts in a given domain before evaluation (of said creativity) inevitably happens in this domain. Hence it can be very problematic to evaluate without considering the application, even though we state that the presence (or level) of creativity is not dependent on the given domain. This entanglement of evaluation and domain knowledge results from the definition of CC, and the definition of creativity in general. As long as CC is defined in terms of ‘behaviour deemed to be creative’ we are relying on an adjudication of actions (behaviour) in comparison to an ill-defined concept (creativity). Without any further specifics we automatically create and evaluate systems in our preferred domain. Should a definition also make some reference to an intuitive leap, or creative step in terms of the abstraction or emergence of a new idea from knowledge already obtained? Even if we did consider incorporating this into a definition — how would one measure such a creative step?

CC research is undertaken within a broad range of subject areas. Attempting to limit this by, for instance, suggesting all research should be conducted only in certain domains would be counter-productive to progress. For an individual researcher to switch application domain may involve a steep learning curve in developing new expertise before any experimental progress could be made. If we state that the presence of creativity is not dependent on working within one application domain then this would appear to be a waste of time, a dismissal of many bodies of work and it would strangle the work of many prominent researchers within the field. Furthermore, as creativity itself remains an ill-defined concept, restricting the areas in which it is studied could hinder development in some unknown way. It is still possible that we could learn more about creativity through one application over another. Hence, we consider it to be beneficial to keep considering more applications rather than less — while focussing evaluation on the system, rather than the product created through representation within the given domain. In this study we only considered papers published at ICC3; many relevant papers have been published elsewhere. Multiple journals have had special issues on CC and there are many other conferences, workshops and journals that look at computational aspects of specific domains such as music, art or design. While a fully comprehensive review of the field would consider all such events, such a review would be infeasible. We have chosen ICC3 papers as a representation of the field as a whole.

The field of Artificial Intelligence (AI) has witnessed a similar focus on application-based systems leading to the development of Weak (or narrow) AI instead of Strong (or general) AI. Many high profile successful instances of Weak AI have made headlines in recent years such as Deep Blue

(Campbell, Hoane, and Hsu 2002) or AlphaGo (Silver et al. 2016). These systems are highly impressive in beating world class humans at a specific task and gain media attention and prestige to their programmers. This may benefit the reputation and status of the field in general, yet these systems do not exhibit or possess a General AI that can tackle multiple different problems as a human would. Domain-specific CC systems are falling into the same single-application trap. A music system hailed as ‘creative’ will not recognise the creativity in a joke unless it is also programmed to recognise the humour representation *and* possesses an ability to recognise general creativity — a term we are still struggling with. An ideal general creative system would be able to generate or appreciate a creative act regardless of domain, yet as with general AI, there is no such system at the moment. The comparison between AI and CC is a natural one if we consider creativity as a feature of human intelligence, but CC should not be labelled as mere application within AI. CC remains a field in its own right as long as we ensure that the questions considered are not limited to the description of creative applications. Such a topic is better described as Creative AI, and does tend to involve more aesthetic endeavours such as the generation of art or images. The focus of the field of CC has always been on developing an understanding of what it means to be creative and how we can emulate this creativity autonomously in computational systems.

Future Steps

It was noted above that there is a lack of papers on scientific and logical problems. As so many early studies in creativity did consider such problems to be relevant (or even fundamental) to creative thinking, it would appear that this indicates a potential gap in the field. Addressing this would require some act to entice researchers to undertake research in one of these areas. The proposal of an annual problem-solving competition to coincide with the annual conference could potentially address such an issue. Similar open competitions have been incorporated into other conferences, for instance the ‘Humies’, a human-competitive event held annually at GECCO (Humies 2017). Such a competition should require the development of an autonomous system that solves a specific logical or scientific problem. As the application domain is set, each system would be adjudicated on the creativity it displayed in its approach to the problem. While such a competition may require some organisation, it would encourage development within this domain.

Much early work in creativity was based on symbolic AI, with representations that could be grasped and understood by the user. In more recent years, applied computing research (including CC) has moved towards a more statistical, machine learning approach. The lack of explanation offered by such systems may have influenced the move away from logical problems towards more subjective, aesthetic problems. Even so, we have a responsibility to communicate the possibilities of what can be achieved through CC more clearly to a general computer science and research audience. Personal experience has indicated that many researchers familiar with machine learning or data science are under the assumption that research in CC always involves either music

or art. It is unfortunate that even among experienced, applied computer scientists, the use of the term *Creative* still translates to aesthetic or artistic. The more fundamental meaning of Creativity and the possibilities that can be reached through proper understanding and research needs to be better portrayed as the field develops.

Developing systems that tackle real-world problems could attract more funding for our own areas of research, either through industrial relations or for academic funding proposals. As an applied computer science area we should always bear in mind that good problem solving requires creativity. Real-world problems such as those proposed in (McCaffrey and Spector 2011) or the propositions for managing Dementia Care (Zachos and Maiden 2013) would be of great interest to the public in general. Acquiring knowledge and developing systems to benefit society and the world around us is surely the ultimate goal of any scientific research; arguably we have a moral responsibility to encourage the development of solutions in such areas in any manner possible.

Conclusion

We have presented a review of application domains considered throughout annual CC events over the past 12 years. By concentrating on the applications considered, rather than the overall purpose of the papers, we hoped to gain some insight as to which topical domains are typically used in discussing the subject of CC. This paper focussed entirely on the domains discussed in publications in the field, while simultaneously stating that the presence of creativity is not dependent on any given domain. Although we would like to state that this is because creativity is ‘domain-independent’, at the moment we would just state that this is because the presence of creativity is not determined by the given application domain. We note that papers based on NLP are continuously well-represented across years. Conversely we note a lack of studies based on logical or scientific problems. Tackling scientific, logical or realistic issues could help bring the reputation of CC away from a purely aesthetic domain towards developing solutions for real world problems.

It is difficult at this time to predict how the field will progress in the coming years, but if the current level of growth is to continue, one can assume CC will become increasingly important field within applied computer science. It is imperative that the field remains balanced as it grows and that we remember to reflect on all areas of growth. As a computational field, a number of autonomous systems for analysing papers in the field are emerging such as Dr Inventor (O’Donoghue et al. 2015) that considers the relationships between studies and the system proposed in (Pollak et al. 2016). As such analytical systems are developed, we must ensure to take a step back to consider the implications of the results obtained, what this may tell us about the field and how we can use this information to shape the development of the field as it progresses.

Acknowledgments

We would like to thank members of the ACC for personal correspondence in collecting this data, in particular Raphael

Pérez y Pérez, Dan Ventura, Amilcar Cardoso, Geraint Wiggins, Pablo Gervás, Dragana Miljkovic and Vincent Corruble. This work is part of the App'Ed project funded by Science Foundation Ireland under grant 13/IA/1850.

References

- Boden, M. A. 1998. Creativity and artificial intelligence. *Artificial Intelligence* 103(1):347–356.
- Boden, M. A. 2004. *The creative mind: Myths and mechanisms*. Psychology Press.
- Butnariu, C., and Veale, T. 2006. Lexical combinatorial creativity with “gastronaut”. In *The Third Joint Workshop on Computational Creativity, 17th European Conference on Artificial Intelligence, ECAI. Riva del Garda: Università di Trento*.
- Campbell, M.; Hoane, A. J.; and Hsu, F.-h. 2002. Deep blue. *Artificial intelligence* 134(1):57–83.
- Cardoso, A.; Veale, T.; and Wiggins, G. A. 2009. Converging on the divergent: The history (and future) of the international joint workshops in computational creativity. *AI Magazine* 30(3):15.
- Colton, S.; Llano, M.; Hepworth, R.; Charnley, J.; Gale, C.; Baron, A.; Pachet, F.; Roy, P.; Gervás, P.; Collins, N.; et al. 2016. The beyond the fence musical and computer says show documentary. In *Proceedings of the International Conference on Computational Creativity*.
- Colton, S.; Wiggins, G. A.; et al. 2012. Computational creativity: the final frontier? In *ECAI*, volume 12, 21–26.
- Confalonieri, R.; Corneli, J.; Pease, A.; Plaza, E.; and Schorlemmer, M. 2015. Using argumentation to evaluate concept blends in combinatorial creativity. In *Proceedings of the Sixth International Conference on Computational Creativity*. Brigham Young University.
- Cos, M. A.; y Pérez, R. P.; Altas, C. L.; Hidalgo, M.; Llera, A. A.; Filosóficas, I.; and de México, A. 2007. A generative grammar for pre-columbian artistic production: The case of el tajín style. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, 39.
- Costa, D.; Oliveira, H. G.; and Pinto, A. M. 2015. ‘In reality there are as many religions as there are papers’ - First steps towards the generation of internet memes. In *Proceedings of the Sixth International Conference on Computational Creativity*, 300.
- Grace, K., and Maher, M. L. 2015. Specific curiosity as a cause and consequence of transformational creativity. In *Proceedings of the International Conference on Computational Creativity June*.
- Humies. 2017. 14th Annual “Humies” Awards for human-competitive results produced by genetic and evolutionary computation. <http://gecco-2017.sigevo.org/index.html/Humies>.
- Johnson, C. G. 2012. The creative computer as romantic hero? computational creativity systems and creative personae. In *Proceedings of the third international conference on computational creativity*, 57–61.
- Jordanous, A. 2011. Evaluating evaluation: Assessing progress in computational creativity research. In *Proceedings of the second international conference on computational creativity (ICCC-11). Mexico City, Mexico*, 102–107.
- Kantosalo, A.; Toivanen, J. M.; Xiao, P.; and Toivonen, H. 2014. From isolation to involvement: Adapting machine creativity software to support human-computer co-creation. In *Proceedings of the Fifth International Conference on Computational Creativity*, 1–8.
- Martins, P.; Pollak, S.; Urbancic, T.; and Cardoso, A. 2016. Optimality principles in computational approaches to conceptual blending: Do we need them (at) all? In *Proceedings of the Seventh International Conference on Computational Creativity*.
- McCaffrey, T., and Spector, L. 2011. How the obscure features hypothesis leads to innovation assistant software. In *Proceedings of the second international conference on computational creativity*, volume 1, 120–122.
- McDermott, J.; White, D. R.; Luke, S.; Manzoni, L.; Castelli, M.; Vanneschi, L.; Jaskowski, W.; Krawiec, K.; Harper, R.; De Jong, K.; et al. 2012. Genetic programming needs better benchmarks. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, 791–798. ACM.
- Morris, R. G.; Burton, S. H.; Bodily, P. M.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *Proceedings of the 3rd International Conference on Computational Creativity*, 119–125.
- O’Donoghue, D. P.; Abgaz, Y.; Hurley, D.; and Ronzano, F. 2015. Stimulating and simulating creativity with dr inventor. In *Proceedings of the Sixth International Conference on Computational Creativity*. Brigham Young University.
- Pagnutti, J., and Whitehead, J. 2015. Generative mixology: An engine for creating cocktails. In *Proceedings of the Sixth International Conference on Computational Creativity*, 212.
- Pollak, S.; Boshkoska, B. M.; Miljkovic, D.; Wiggins, G. A.; and Lavrac, N. 2016. Computational creativity conceptualisation grounded on iccc papers. In *Proceedings of the Seventh International Conference on Computational Creativity*.
- Shao, N.; Murali, P.; and Sheopuri, A. 2014. New developments in culinary computational creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*.
- Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.
- Ventura, D. A. 2008. A reductio ad absurdum experiment in sufficiency for evaluating (computational) creative systems. In *Proceedings of the 5th international joint workshop on computational creativity*. Computational Creativity.
- Zachos, K., and Maiden, N. 2013. A computational model of analogical reasoning in dementia care. In *Proceedings of the Fourth International Conference on Computational Creativity*, volume 48.

Stay Awhile and Listen to 3Buddy, a Co-creative Level Design Support Tool

Pedro Lucas and Carlos Martinho

INESC-ID and Instituto Superior Técnico, Universidade de Lisboa
Av. Prof. Doutor Cavaco, Silva, 2744-016 Porto Salvo, Lisboa, Portugal.
{pedro.lucas, carlos.martinho}@tecnico.ulisboa.pt

Abstract

There is untapped potential in having a computer work as a *colleague* with the video game level designer as a source of creative stimuli, instead of simply working as his slave. This paper presents 3Buddy, a co-creative level design tool exploring this digital peer paradigm, aimed at fostering creativity by allowing human and computer to work together in the context of level design, and describes a case study of the approach to produce content using the Legend of Grimrock 2 level editor. Suggestions are generated and iteratively evolved by multiple inter-communicating genetic algorithms guiding three different domains: innovation (exploring new directions), guidelines (respecting specific design goals) and convergence (focusing on current co-proposal). The interface allows the designer to orient the tool behaviour in the space defined by these dimensions. This paper details the inner workings of the system and presents an exploratory study showing, on the one hand, how the tool was used differently by professional and amateur level designers, and on the other hand, how the nuances of the co-creative interaction through an intention-oriented interface may be a source of positive influence for the creative level design process.

Introduction

Creativity is of paramount importance in our current day and age, and more rewarded than technical prowess in certain domains, in particular in the games software industry (Murphy-Hill, Zimmermann, and Naggapan 2014). While no consensus exists regarding the definition of creativity, there is an overall agreement that *creativity* can be seen as the “interaction among aptitude, process and environment by which an individual or group produces a perceptible product that is both novel and useful as defined within a social context” (Plucker, Beghetto, and Dow 2004). This work explores how a computer could foster creativity in an important component of digital game development: Level Design.

Computational Co-creativity

Most people face computers as facilitating tools, and disregard their potentially valuable contribution to any creative process, as if something impossible for a computer to help with. The fact that creativity is often considered to be a mental process occurring within an *individual*'s head strengthen this belief when, in fact, creativity is an ability that can be

stimulated in a distributed way within a *group* (Sawyer and DeZutter 2009).

Lubart (2005) identifies four ways a computer can support human creativity: management of creative work, in which the computer takes the role of a “nanny”; communication between individuals collaborating on creative projects, in which the computer takes the role of a “pen-pal”; the use of creativity enhancement techniques, in which the computer acts as a “coach”, and; the creative act through integrated human-computer cooperation during idea production, in which the computer acts as a “colleague” and co-creator. Lubart (2005) classifies the last category as the most ambitious way that computers can support human creativity, but also one of the most interesting to explore, mainly because of the inherent potential of an interaction between computers, that excel at exhausting a search-space, and humans, that excel at transforming rough concepts into coherent ideas. Our work shares this point of view and explores the digital *colleague* paradigm.

Content Creation for Digital Games

Procedural Content Generation or PCG (Togelius et al. 2011), is a growing trend in digital game development, a set of techniques able to quickly generate “correct” content within a set of constraints, and support a more cost-efficient process for content-heavy digital games. Although this technique excels at generating “filler” content, the digital game development process lacks tools to support the design of critical and crucial moments for the game experience, moments that are able to surprise the player and make a game stand out from others on the market (possibly using the same PCG toolbox). To complement the content *generation* tools, we need content *creation* support tools.

We believe *Level Design*, “the thoughtful execution of gameplay into gamespace for players to dwell in” (Totten 2014), is a good example of an activity which would benefit from a computer-assisted co-creative tool. Several modern games are released with content editors and modding tools, facilitating the creation of custom content, although such tools are generally the ones that supported the creation of the base game itself and offer little to help the user create genuinely interesting and novel content that could be classified as “creative” and have a value on its own. We need tools to help achieve that goal.

Co-creative Level Design Support Tools

Our research builds on previous work in mixed-initiative co-creative support tools such as the Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2013) and Tanagra (Smith, Whitehead, and Mateas 2011), and takes the research one step further in terms of the digital *peer* paradigm. Our design is guided by the seminal work on *Lateral Thinking* by DeBono (De Bono 1977) and the experiments by Yannakakis et al. with *mixed-initiative co-creativity* (Yannakakis, Liapis, and Alexopoulos 2014) merging both Lateral Thinking, which use the interaction should facilitate but not enforce, and *Diagrammatic Reasoning* (Vile and Polovina 1998), i.e. reasoning via the use of visual representations, a cognitive process inherently present in level design.

This work aims at conceiving a support tool that will *proactively* take part in a dialogue with a designer during the level design process that will (1) benefit the *creative process* per se, (2) allow to reach more *creative outcomes* as an output of this process, and; (3) have a lasting effect on the participants own *creative abilities*. Yannakakis et al. (Yannakakis, Liapis, and Alexopoulos 2014) point there are two types of evaluation when a computational creator is involved: the evaluation of the intermediate and final outcomes and the evaluation of the co-creative process. We believe the impact of the interaction on the *participants'* (both human and digital) creative capabilities to be of importance when evaluating creativity support tools.

This work is framed within the field of computational creativity, a key issue being to understand how the interaction with our level design tool could help support a more creative process, outcome, and impact the abilities of both participants. Our work contributes towards the enhancement of human creativity with the aid of computer programs, framing it as a Creativity Support Tool (CST) (Shneiderman 2007).

In this paper, we present an *exploratory study* probing how different participants, with different game and level design backgrounds, interact with our creative support tool in the context of a specific level design task for a commercial video game, and how it impacts their approach to the task. The remaining of the paper is organized as follows. We start by describing the commercial game level design tool used in this work, as well as the metaphor for interacting and communicating with our CST and detail its implementation. Then, we present our user exploratory studies and discuss the results of this evaluation. We conclude with our final remarks and directions for future work.

Legend of Grimrock 2 Editor

The game Legend of Grimrock 2 (Almost Human Ltd. 2014), henceforth designated by the acronym LoG2, is a computer game of the dungeon crawler genre which includes an in-game level editor. The LoG2 editor (Figure 1) provides the designer with a 2D canvas, as well as a series of tools to edit the layout of the level as well as its content. It is also possible to test, in real-time, the impact of the changes performed on the level by running the game in one of LoG2 editor viewports. In this paper, the LoG2 editor serves as the primary editing tool for the level designer, whereas our

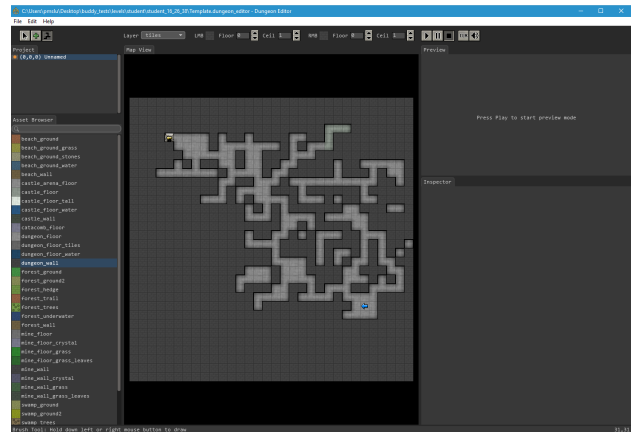


Figure 1: Legend of Grimrock 2 editor

companion tool will propose suggestions related to the content being created inside the LoG2 editor. We also focus on the co-creation of the dungeon layout, which in LoG2 is represented by a grid and is available as a plain text Lua script file, that serves as the communication point between the editor and our companion tool.

Editor Buddy (3Buddy)

Editor Buddy, henceforth designated *3Buddy*, is a C# application that runs independently but alongside the LoG2 in-game editor. Its primary goal is to foster creativity during a level design activity by presenting visual hints and suggestions iteratively, while the designer works inside the game editor. Following a dialogue metaphor, its interface provides the level designer with visual information as a way to stimulate creativity, and guides the application behaviour to adapt to the moment-to-moment needs of the creative process.

3Buddy Interface

The interface of 3Buddy is inspired by the control knobs found on analogue synthesizers, that shape the sound produced by these musical instruments. In our case, the control sliders (sliders were found to be preferred to control knobs during early prototyping) shape the direction of the dialogue with the level designer during interaction. 3Buddy user interface (Figure 2) essentially consists of three control sliders and an interactive canvas. The control sliders are:

- a slider controlling the amount of innovation expected in the current moment of the creative process (*innovation*);
- a second slider controlling the importance of complying to specific design goals (a pane allows the selection of currently active guidelines) (*guidelines*);
- a third slider controlling how much the current solution being edited by the level designer should be taken into consideration when making suggestions (*convergence*).

A simpler version of the interface that transforms all sliders into switches that can simply be turned on or off (Figure 2 middle) is also available to the level designer. Usability

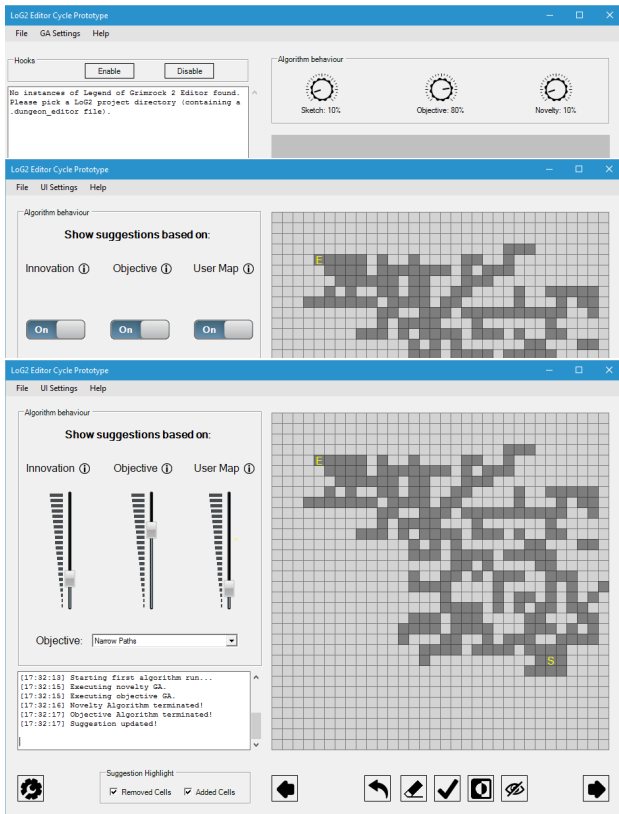


Figure 2: 3Buddy interface at different stages: (top:) early prototyping with knobs (cut) (middle:) standard mode with switches (cut), and (bottom:) expert mode with sliders (full).

testing encouraged us to include this option as it is useful on the first interactions to get a feel for the individual impact of each control slider on the overall co-creation process.

The interactive canvas supports the *dialogue* between 3Buddy and the level designer. On the one hand, the designer can select sections of the idea to import into the editor or define the focus in terms of the interaction with the creative companion. A set of editing tools similar to the ones provided in LoG2 editor are available to the designer to allow for the precise selection of specific parts of the dungeon layout. On the other hand, 3Buddy presents its suggestions in terms of colour-coded differences to the current version of the level layout present in the LoG2 editor, i.e. additions and removals are presented in different colours. The colour coding can be turned on and off, which is useful when a commitment is reached. Although 3Buddy always works in a complete solution, it only shows changes in the part of the dungeon that is the current focus. The designer, however, can ask 3Buddy to reveal “hidden” parts of the layout at any time. Finally it is possible to revisit past suggestions. Overall, the interactive canvas allows to intuitively and visually communicate both focus and suggestions to support diagrammatic reasoning, a cognitive process inherently present in level design.

3Buddy Behaviour

3Buddy’s behaviour is triggered each time the dungeon layout is saved by the level designer in LoG2 editor or after an inactivity time-out. As a response to this trigger, a suggestion is computed and presented on the interactive canvas of 3Buddy. The suggestion can be ignored by the designer or part (or the whole) of it integrated in the current level layout. The designer can also select parts of the canvas to specify the section on which both the designer and 3Buddy are working at the moment. This quick interaction cycle guided by the three sliders is what distinguishes our approach from previous ones. By allowing the designer to consider and evaluate intermediate steps in a less rigid structure, we expect to tap into a process with a greater creative potential.

Computational Approach. To compute a suggestion and present it to the level designer, 3Buddy conceptually uses three different base sets of suggestions (Figure 3):

- a set of suggestions that are computationally evolved to be close to the current dungeon layout the designer is working on in LoG2 editor (*convergence* pool).
- a set of suggestions that are computationally evolved to be radically different from what the designer is currently working in the LoG2 editor (*innovation* pool).
- a set of suggestions that are computationally evolved to comply to the currently active design goals and guidelines (*guidelines* pool).

The innovation and guidelines pool are initialized with random individuals and evolved according to their respective purpose, while the convergence pool is initialized with copies of the current dungeon layout that will typically undergo small changes through mutation.

A new population (*suggestion* pool, see Figure 3) is then created by randomly selecting individuals from each one of the three base sets, according to the proportions defined by each one of the control sliders (convergence, innovation and guidelines). The sliders metaphorically control the flow of ideas from each base set to the suggestion pool that will give birth to 3Buddy’s suggestion.

This new population now constitutes the guidelines pool and a new cycle begins: the convergence pool is updated with new individuals representing the current dungeon layout in LoG2 editor and evolved to create proposals that are small variations from it; the innovation pool is evolved to create individuals moving away from the current proposal, and; the guidelines pool, a mix of the previous three base suggestion sets, is evolved to find a new dungeon layout to suggest to the designer guided by the active design goals. Once selected, the new suggestion is presented on the interactive canvas, highlighting the differences to the current layout and hiding all details that are not in the area currently under focus. 3Buddy then waits for the next trigger before starting a new cycle.

Implementation. The behaviour of 3Buddy is guided by genetic algorithms (GA) associated with each set, which themselves are tied to the dungeon section on which both 3Buddy and the designer are working at the moment.

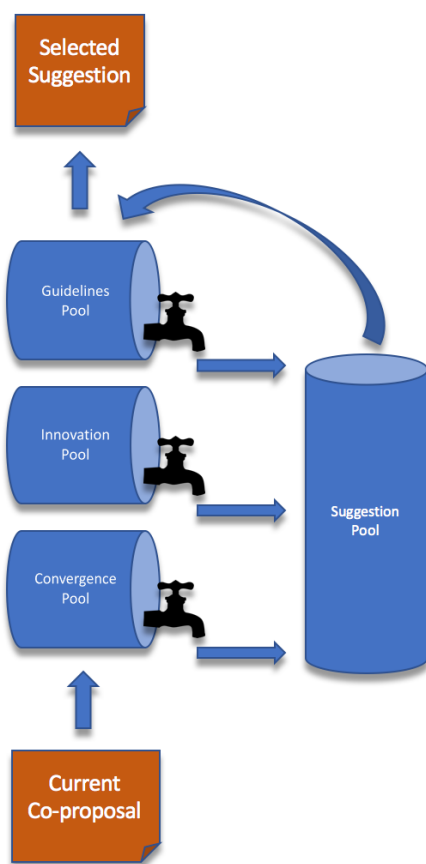


Figure 3: 3Buddy computational approach

The GA implementation is based on the Genetic Algorithm Framework by Newcombe (2015) which was adapted to account for the bi-dimensional grid layout of the dungeon. Each dungeon layout suggestion (a 32x32 square grid) is represented as an unidimensional line-major chromosome (Figure 4, middle), in which each gene records whether it is crossable or not. For this exploratory evaluation, each GA population was composed of 30 individuals, that are initially randomly generated, with approximately 50% of crossable tiles, and 80 generations were computed each time 3Buddy's behaviour triggers the evolution of a suggestion set.

The parents for the next generation are selected using roulette wheel selection, i.e. the probability of being selected is proportional to the fitness of the individual. Crossover occurs with a 80% probability. The standard crossover operator was redefined to take into account the bi-dimensional nature of such space: conceptually, a square (in this specific implementation we used a 3x3 square, mapped to the one-dimensional chromosome space) is randomly chosen in the dungeon layout to define the crossover boundary combining the genetic material from two offsprings (see Figure 4, top and bottom).

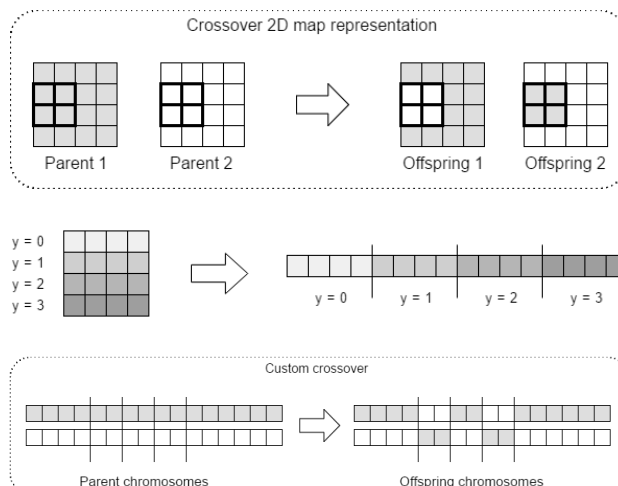


Figure 4: (top:) Example of our crossover approach using a 2x2 square and in a 4x4 grid, (middle:) Example mapping of a 4x4 space to a 16-gene chromosome, and (bottom:) Example mapping of a 2x2 square crossover to a 16-gene chromosome.

Replacement policy uses generational replacement, where the best offspring replaces its least fit parent. Mutation has a 15% chance of occurring and consists in swapping two genes within the chromosome. Finally, an elitism policy was used: the top 5% individuals from a population are copied to the next generation without being modified.

To account for their different purposes, each GA used a different fitness function:

- The “convergence” GA uses the number of equal genes between two chromosomes as the fitness function.
- The “innovation” GA uses the number of different genes between two chromosomes as a measure of fitness.
- The “guideline” GA and “final suggestion” GA use a linear combination of fitness functions specific to each active design goal.

The design goals implemented in 3Buddy for the evaluation task were: “a path exists between an entry tile and an exit tile”; “the dungeon has narrow halls”; “the dungeon has ample rooms”. Each design goal has a specific fitness function related to that goal. Changing the active design goals resets the “guidelines” suggestion pool with a new random set of chromosomes that are evolved according to the combination of the new active design goals.

User Study with 3Buddy

To evaluate the perceived utility and efficiency of 3Buddy in the context of level design, we conducted a user study performed on a small group of participants for short periods of time, inspired on a realistic task for a game development studio. By *utility*, we refer to the ability to contribute, direct or indirectly, with useful content – the definition of usefulness remaining at the designer’s discretion. By *efficiency*, we refer to the possibility of configuring 3Buddy in such a way

as to produce coherent and useful content according to the needs and intentions of the designer at a given time during the co-creative process.

The study included two tasks designed to explore different contexts of use for 3Buddy in level design: the first task targeted content creation from scratch while the second task focused on modifying content previously created.

Sample: 6 participants were selected for this study, with ages ranging from 21 to 35 years old ($M = 27.3$, $SD = 5.16$, 6 male). We used a purposeful sampling method and selected participants according to two different profiles based on background and expertise: 3 professionals or expert individuals, game or level designers, and 3 amateur or inexperienced individuals, Computer Science MSc students from a Games course. All the participants expressed having a clear interest in Level Design.

Procedure: Participants would arrive individually at the laboratory where the study would take place and be introduced to the most important features of the game Legend of Grimrock 2, its editor and 3Buddy, through a 5 minute video tutorial. They would then experiment with these tools and freely ask questions regarding their usage to the researcher supervising the study. The whole introduction / tutorial took around 30 minutes.

After this initial contact, participants would be asked to design a new playable dungeon floor between two existing floors (fictitious and not provided) with certain constraints: they would be using the LoG2 editor while interacting with 3Buddy to support them during the level creation process; they would have to design a solution with a valid path between the (unmovable) entry and exit points of the level that represent the access to contiguous levels (Figure 5), and; the solution should be as interesting as possible for the player.

This first task had an additional constraint:

The level layout should be maze-like and contain narrow paths, as monsters with a charge attack, that will be placed later in the development, should be able to take full advantage of their ability in this level.

The participants were then told they would have all the time needed until they would feel totally satisfied with their solution. However, after 20 minutes (the average duration of this task, as measured during pre-test with 5 other participants), we would ask him to interrupt his current work and start working on a second task. If the task was completed before that time, we would interrupt the participant as he would stand up to inform the researcher. This new task is in all similar to the first except for an additional constraint:

The level layout needs to accommodate for challenging big monsters that will be placed on this level but are too big to fit the standard narrow halls.

This second task had no time limit, but took around 20 minutes to complete. After the level layout is finished, we asked the participant to fill a questionnaire (circa 10-15 minutes) and undergo a semi-structured interview (circa 20-30 minutes). The last stage of the study consisted in looking at the solution provided by the participant, congratulating him

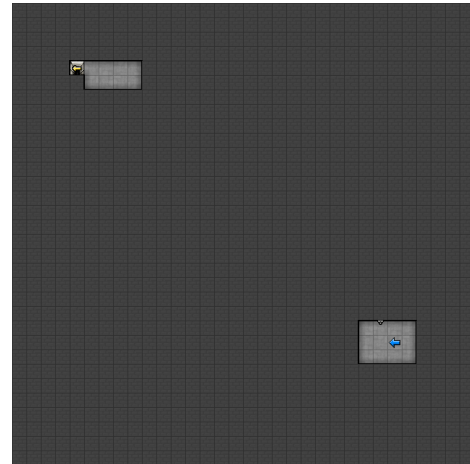


Figure 5: Starting level layout for evaluation

for his accomplishment, thanking him for his participation and saying goodbye. The full experiment had an average duration of 2h per participant.

Data collection: To evaluate 3Buddy, we used qualitative data gathered through participant observation, a questionnaire, and a semi-structured interview.

While a researcher was always present in the back of the room, taking notes and answering technical questions when requested to by the participants, observation was complemented by screen recording during the entire process: we were interested in the participants' actions during level design activities including their interaction with both the in-game editor and 3Buddy.

We used a linear scale questionnaire to get a measure of usability for the LoG2 editor and the 3Buddy interfaces. The questionnaire also measured the discrepancy between the participant expectation when tuning the control sliders in a certain configuration when compared to the associated suggested content by 3Buddy. Finally, the questions incentivised the participant to think about certain issues that would be explored during the following interview.

After participants were finished with the questionnaire, a semi-structured interview was conducted. The goal for the interview was to draw out patterns from common concepts and insights regarding the personal experience of each participant while interacting with 3Buddy.

Results

Questionnaire

The following agreement statements (5-point Likert scale) evaluated the usability of LoG2 editor and 3Buddy: (U1:) it was easy to edit the layout of my level using the LoG2 editor; (U2:) it was easy to configure 3Buddy's behaviour using the available interface controls; (U3:) it was easy to make and edit a section of the map suggestion made by 3Buddy; (U4:) there were no issues regarding the communication between the LoG2 editor and 3Buddy.

Participants reported 3Buddy to be easy to use and its integration with LoG2 editor intuitive and satisfying. This was reinforced by observation and interviews. Table 1 shows the results from the usability section of the questionnaire.

5-point Likert	U1	U2	U3	U4
Totally disagree	0	0	0	0
Somewhat disagree	0	0	0	0
Neither agree nor disagree	0	0	1	1
Somewhat agree	2	2	1	2
Totally agree	4	4	4	3

Table 1: Frequency of answers to usability questionnaire

Regarding the evaluation of 3Buddy’s behaviour, Figure 6 shows how 3Buddy’s suggestions, when using the *innovation* control slider, were as expected and how useful they were perceived by the participants. Figures 7 and 8 do the same for the *guidelines* and *convergence* control sliders.

The experiment confirmed what we had learned during pre-test: suggesting the participant to start with standard mode before moving to expert mode is key to the creation of adequate expectations regarding the interaction.

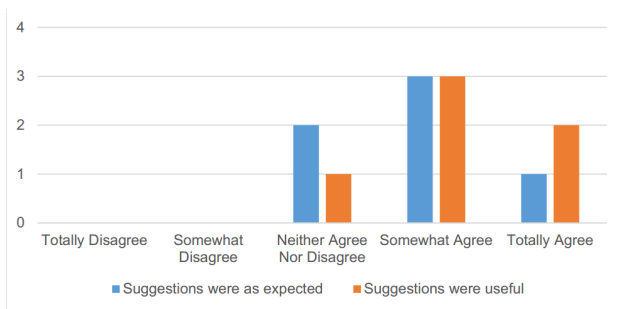


Figure 6: Frequency of answers to (B1:) suggestions using “innovation” were in line with my expectations (B2:) suggestions using “innovation” were useful.

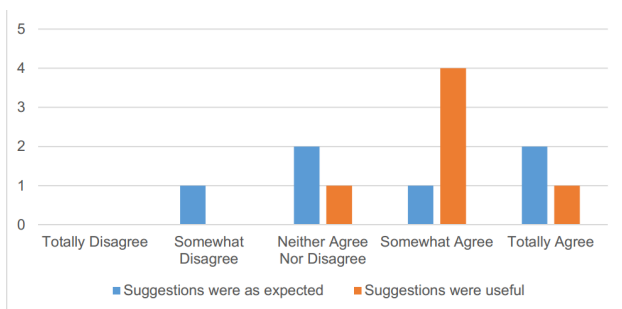


Figure 7: Frequency of answers to (B3:) suggestions using “guidelines” were in line with my expectations (B4:) suggestions using “guidelines” were useful.

The exploration of the negative marks showed that the “guideline related suggestion not as expected” (Figure 7)

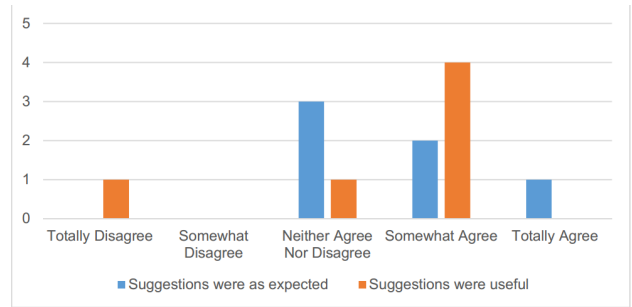


Figure 8: Frequency of answers to (B5:) suggestions using “convergence” were in line with my expectations (B6:) suggestions using “convergence” were useful.

was actually perceived as a good thing, “a pleasant surprise”, and the 1-rating in convergence control (Figure 8) was related to the inability of 3Buddy to understand what the level designer was “thinking and trying to do rather than doing”. This particular participant also reported “not needing the help of a tool to be able to perform his work”. This never happened with the amateur level designers who were thrilled with the help provided. This is a factor that clearly needs further study, as a tool aimed at helping in the creative process could be perceived as a competitor in a domain where creativity is more valuable than technical skills such as the digital games industry.

Observation

Through direct observation and screen recording, we were able to identify key-events and key-issues during participant interaction with 3Buddy, as well as important processes which occurred over time. Identified *key-events* were:

- Increase in interaction with the behaviour control sliders, immediately after the user changed to expert mode;
- Increase in interaction with 3Buddy canvas and suggestion selection in all cases after moving to expert mode;
- Exporting suggestion selections almost always meant a consequent refinement of that content in the designer’s working level layout;
- When asked to work towards a different objective with 3Buddy, the vast majority immediately changed 3Buddy guidelines as a response.

Key-issues detected through observations:

- When facing participants with poor suggestions from 3Buddy that rarely translated into an attempt to reconfigure its behaviour using the standard mode;
- When trying to preview the entire suggestion, the majority of participants would perform a selection of the whole canvas instead of using the dedicated visibility button;
- Participants almost never interacted with the history buttons to revisit previous suggestions;
- After changing to expert mode, participants never changed back to the standard mode.

Important processes such as decision making or control interaction were essentially different for each participant, but here are the more frequent patterns detected:

- Professionals had a predetermined idea for the level and only interacted with 3Buddy after they were finished with a first version of the layout;
- Non-professionals started interacting with 3Buddy early on and effectively performed selections, exported and improved these changes frequently during the interaction;
- The interaction with the behaviour switches (standard mode) and/or sliders (expert modes) increased over time;
- Participants would perform selections over the canvas regularly, as part of their creation process, even if the result was not exported;
- No participant felt the need to use the revert button to go back to a previously layout suggestion;
- Although there was no time limit, the duration of each task was very even amongst all participant professional and amateur alike, averaging 20 minutes.

Finally, and as a *heuristic* approach to validate the adequacy of the interaction with 3Buddy in a creative context, observation allowed us to verify that some techniques presented by Debono in his seminal work on Lateral Thinking (De Bono 1977) did occur during the interaction with 3Buddy. Specifically, we identified the following techniques in the recorded interactions: to explore simultaneously different alternatives, to change focus during the interaction, to break free from the requirements and limits imposed by the design process, to allow for the connection of unrelated and random and provocative input to open new lines of thinking, and to harvest the best ideas and reshape them into a practical solution. This suggests the interaction with 3Buddy does not limit the use of such techniques in the creative process.

Semi-structured Interview

The interview was guided by the following script, while exploring aspects raised by the participants:

1. Overall, did you find 3Buddy useful?
2. In which case did it work best: on the first task where you had to start from scratch, or on the second task where you had to rework your current level?
3. If you moved to expert mode, what made you change? If you have not, why did you not change?
4. What combination(s) of behaviour switches/sliders would you recommend a friend using this tool for the first time?
5. What would be the combination(s) you would not recommend?
6. Was there a right time to use a particular switch or slider all the way up or down?
7. If you could make any change to the interface, what would it be? Why?
8. If you could make any change to the behaviour, what would it be? Why?
9. If you could have a mode where 3Buddy suggestions would be directly applied to your work without the need for your approval, how would you feel about that?
10. Any other comment that would help us improving 3Buddy?

Results collected from the semi-structured interviews were more subjective and, thus, harder to convey in their entirety. Some interesting patterns, however, emerged during this process, and specific improvements were suggested.

Generally, non-professionals preferred to use 3Buddy during the first task where they were asked to create content from a minimal template. In opposition, professionals, preferred to use 3Buddy during the second task where they were asked to modify their work towards a different design goal instead of creating content from scratch.

Regarding control configurations, although innovation and convergence switches and sliders were the controls which had most disparity in terms of usage and preference over time, we found there was a relation to both the task and the stage of the design participants were focusing at the time. Generally, participants recommended alternating between innovation and convergence on the full map in the early stages to quickly generate a large quantity of content, and doing the same thing on small portions of the map at later stages of the level design, to explore detailed changes on parts of the map they were unhappy with.

Regarding configurations of controls to avoid, most comments were in line with our expectations: “do not use innovation if you just want to slightly modify your work” and “do not use convergence if you want to explore different alternatives”. These comments strengthened that the function provided by 3Buddy as a result of interface interaction was in line with user expectations.

Suggested interface improvements included:

- Parametrization of the colour coding scheme and small refinements at the interface level to optimize intuitiveness;
- Highlight of viable paths from start to finish, and visual feedback based on compliance to the active guidelines;
- Visual differentiation between older and recent suggestions, e.g. more vibrant colours for newest suggestions;
- An initial procedure to help setting up an optimal configuration according to the type of user.

Suggestions to improve 3Buddy behaviour included:

- A way to ask for an immediate new suggestion without having to make any change in the current dungeon layout;
- Add continuous sliders to allow for greater control on map complexity, rather than using statements for guidelines;
- Add a slider to set up the trade-off between the quality of the suggestions and the response time;
- Add a slider defining how much the suggestions presented could fall outside the region selected by the user, rather than strictly adhering to it;
- 3Buddy should identify patterns in the current layout and be able to replicate this style in the following suggestions.

At first sight, most comments seem to point to a need participants had to be more in control. A closer look, however, revealed that participants actually felt the need to clearly communicate their *intentions* at different moments in the creative process to 3Buddy. And that was exactly what the designers were able to intuitively express through the three sliders provided by the interface.

Conclusions

In this paper, we presented 3Buddy, a co-creative support tool exploring the digital *colleague* paradigm to foster creativity in the context of video game level design. We described 3Buddy's simple and innovative interface composed of: (1) an interactive 2D canvas, allowing to intuitively and visually communicate both focus and suggestions to support diagrammatic reasoning, a cognitive process inherently present in level design, and; (2) three control sliders defining what is important at a certain moment in the creative process, expressed as a combination of three dimensions: innovation, wanting to explore new directions; guidelines, following specific design goals, and; convergence, focusing on the current co-proposal. We explained how 3Buddy was connected to the flow of level creation in the commercial game Legend of Grimrock 2, and how its implementation based on multiple and inter-communicating genetic algorithms supported this approach. Finally, we presented an exploratory qualitative study with both professional and inexperienced designers in the context of two level design tasks: one based on the creation of new content and another based on the modification of existing material. The study supported that, although 3Buddy was perceived as intuitive and useful and that standard lateral thinking techniques were observed during the interaction, professionals and inexperienced participants had very distinct patterns of interaction with the three sliders with which they expressed their intent during the creative process. Such patterns need to be taken into account when creating creativity support tools such as 3Buddy.

Future directions: As a result of the encouraging results obtained from this exploratory study, we are currently exploring several directions to improve 3Buddy. We are exploring how the same (but refined) interface could be used to support other level design specific tasks and are looking at puzzle creation as well as enemy and loot placement within a LoG2 level, as well as how these different dimensions could be integrated into one single CST. We are researching how the interface could be personalized for each user while exploiting patterns learned from the interaction and, as such, be more proactive while warmly accepted by all designer typologies. We are also exploring how different search algorithms colour the type of suggestions provided by 3Buddy. Finally, we want to test the impact of the co-creative process on the final player experience as well as on the impact on the creative posture of the designers themselves.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

- De Bono, E. 1977. *Lateral thinking: a textbook of creativity*. Pelican books. Penguin Books.
- Liapis, A.; Yannakakis, G.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of the 8th International Conference on the Foundations of Digital Games (FDG 2013)*, 213–220.
- Lubart, T. 2005. How can computers be partners in the creative process: Classification and commentary on the special issue. *International Journal of Human Computer Studies* 63(4-5 SPEC. ISS.):365–369.
- Murphy-Hill, E.; Zimmermann, T.; and Naggapan, N. 2014. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *Proceedings of the 36th International Conference on Software Engineering (ICSE 14)*, 1–11. New York, NY, USA: ACM.
- Newcombe, J. 2015. Genetic algorithm framework (<https://johnnewcombe.net/gaf>).
- Plucker, J.; Beghetto, R.; and Dow, G. 2004. Why isn't creativity more important to educational psychologists? potentials, pitfalls, and future directions. *Creativity Research, Educational Psychologist* 39(2):83–96.
- Sawyer, K., and DeZutter, S. 2009. Distributed creativity: How collective creations emerge from collaboration. *Psychology of Aesthetics, Creativity, and the Arts* 3(2):81–92.
- Shneiderman, B. 2007. Creativity support tools: accelerating discovery and innovation. *Communications of the ACM* 50(12):20–32.
- Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):201–215.
- Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.
- Totten, C. 2014. *An Architectural Approach to Level Design*. A K Peters/CRC Press.
- Vile, A., and Polovina, S. 1998. Thinking of or thinking through diagrams? the case of conceptual graphs. In *Proceedings of Thinking with Diagrams Conference*.
- Yannakakis, G.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-initiative co-creativity. In *Proceedings of the Foundation of Digital Games 2014 (FDG 2014)*.

Applying Narrative Theory to Aid Unexpectedness in a Story Generation System

Todd Pickering
School of Computing
University of Kent
Canterbury, Kent, UK
tmp9@kent.ac.uk

Anna Jordanous
School of Computing
University of Kent
Medway, Kent, UK
a.k.jordanous@kent.ac.uk

Abstract

Predictability is the polar opposite of originality, and as such it is a notable obstacle that should be overcome in the pursuit of computational creativity. Accurately modelling a human's understanding of predictability would be a monumental task, requiring a contextually rich network of social interaction, literature, news, and media. However, by artificially instilling a computer with some basic ideas about what is predictable in a given scenario, it can begin to gain an understanding of how to subvert expectation.

This project attempts to implement such a process into a specially designed story generation system known as *Chronicle*, inspired by Vladimír Propp's *Morphology of the Folk Tale*. *Chronicle* aims to fine-tune narrative direction and progression in a system modelled on predictability.

Decisions made during the story generation process are based on probabilities defined by the expectations of the typical reader, and are amassed to formulate an overall predictability rating. The decision making process is manipulated by the system in order to pursue a customisable predictability target.

Chronicle was demonstrably accurate at evaluating its output in some cases, and less accurate in other cases. Further refinement is required to increase its efficacy, but it presents a promising step towards negotiating predictability in computational creativity.

Introduction

Computers are capable of solving mathematical problems due to their comprehensive knowledge of the existing laws of mathematics. If we extrapolate this, we can surmise that computers may also be capable of solving other types of problems, so long as they possess the necessary knowledge, context, and understanding to do so.

However, this presents a significant challenge when negotiating computational creativity in written fiction, since there are many facets of creativity and authorship that cannot be so rigorously defined.

One such facet is the way in which a narrative is constructed: how one event follows another, and how this impacts future events. It demonstrates an understanding of cause and effect, contextual awareness, and a capability to

make decisions, all of which are fundamental steps towards truly creative computing.

Objectives

The project objectives are as follows:

- Create a functional story generation system capable of producing a variety of outputs.
- Implement narrative theory during development to aid the generation process.
- Ensure that the system is able to affect narrative direction and progression based on its understanding of predictability.

The Importance of Unpredictability

Computers struggle to display any degree of spontaneity or unexpectedness due to the foundation of rules and constraints upon which they are built. This is advantageous for mathematical and scientific pursuits, where consistency and reliable behaviour are paramount. However, it becomes an issue wherever computational creativity is concerned, due to the disparity between predictability and originality. This is doubly the case when fiction is involved; a story with a predictable outcome can lead to an unsatisfying experience for the reader.

It is this intrinsic predictability that must be overcome in order to generate truly creative output. The first logical step is to make the computer aware of when it is and is not being predictable.

Existing Work

Computational creativity in written fiction has been explored and investigated in numerous ways since the early 1970s. However, it is the more recent developments that demand a greater focus, since they tend to address specific concepts and issues in greater and more relevant detail.

Unexpectedness

Kazjon Grace and Mary Lou Maher established that "unexpectedness is [...] a vital component of computational creativity evaluation" (Grace and Maher, 2014). They proposed a series of five properties with which to standardise expectations - "predicted", "prediction", "scope",

“condition”, and “congruence” - and six dichotomies which “[categorise] creativity-relevant expectations based on these properties”.

This model notes the salient differences between variegated degrees of expectation, and in doing so highlights some of the trickier aspects that should be negotiated. For instance, “prediction” requires an accurate numeric representation of the potential variance in values for a given expectation. Using an example from Grace and Maher’s paper, this is relatively simple to apply when defining an expectation of an object’s height as somewhere “between two and five metres”. But when considering the expectation of progression of story events, it becomes rather more difficult to numerically assimilate a reader’s opinion in quite the same way. Unfortunately, we often have to compromise and approximate subjective opinions in this manner (as is the case with *Chronicle*).

Due to the difficulty in translating these existing concepts, it would be beneficial to develop a new system with which to establish and monitor unexpectedness.

A computer must have a degree of awareness about what constitutes predictability before it can begin to overcome it. We can understand ‘predictability’ as a measure of how likely something is to happen. In that respect, it is closely linked to probability, which is an ideal concept to utilise considering its capacity to formalise chance and likelihood into hard data.

Every time a decision needs to be made during the story generation process, each of the possible options can be assigned a probability based on the typical reader’s expectation. When a decision is made, the system can record the probability of the chosen option, and begin to gain an understanding of the overall predictability of the story.

Narrative Theory

Narrative theorists typically analyse fictional constructs for literary purposes, yet the act of breaking down a structure into its constituent parts to gain a better understanding of its composition is an intrinsically mechanical process. As such, the work of narrative theorists has an enormous potential relevance to computational creativity.

Vladimir Propp analysed one hundred Russian folk tales, and in doing so identified a series of recurring elements (Propp, 1968). He established seven character archetypes (or *actants*), and a series of thirty-one story events (or *functions*). Propp determined that each of the stories he analysed was comprised of a combination of these actants and functions. Specific details about these actants and functions varied from tale to tale, but the underlying structure remained largely the same.

For instance, in one crucial story function, a nefarious character commits an act of villainy. The details of this function could be fantastical, with a wicked witch or an evil dragon casting a spell or abducting a person. Instead, the details may be grounded in reality, with an odious stepmother or a belligerent neighbouring tsar ordering a murder. The specific details are not especially important; what matters is that an evil act occurs in order to upset the equilibrium and motivate the hero’s quest.

An argument could be made that Propp’s theory is a reductionist view of narrative structure, and therefore too restrictive for a story generation system; it is certainly true that not all fiction is comprised of these actants and functions. But logistically speaking, the computer has to be provided with a structure of some description, and from an engineering perspective, Propp’s theory lays an ideal foundation for formalisation into software with a reasonable degree of freedom in its potential output.

Pablo Gervas sought to employ Propp’s *Morphology* to create new narrative structures (Gervas, 2013). Gervas accomplished this by incorporating Propp’s functions and actants in fabula and plot driver generators.

However, Gervas’ research is not without its issues. He suggests that “some deviation is allowed [...] by shifting certain character functions to other positions in the sequence”, while Propp’s theory states that “the sequence of functions is always identical” (Propp, 1968). As such, Gervas’ use of plot driver generators to alternate the order of functions appears to deviate from Propp’s literature.

Gervas revisited Propp with another research project focusing on plot structure (Gervas, 2016), which relies heavily upon his aforementioned 2013 project. A notable addition relates to what Gervas names “long-range dependencies”, which are the links established between corresponding story functions, such as a character being kidnapped, and the same character being rescued later in the story. Gervas notes that these “long-range dependencies” have had a “very significant impact on the quality of the resulting plots”.

Narrative theory can undoubtedly be used to aid the structure of story generation systems, but its efficacy can be affected by the degree of faithfulness to the original literature, and an evaluative process should be employed since the implementation of theory is not infallible.

Vladimir Propp’s *Morphology of the Folk Tale* (Propp, 1968) presents itself as an ideal framework, since each of its constituent parts can be assimilated into computerised functions. This would not necessarily result in as rigid a structure as one might first anticipate; while “the sequence of functions is always identical”, Propp stated that “by no means do all tales give evidence of all functions” (Propp, 1968). In other words, despite the fact that story functions are never rearranged, there is the potential for a number of them to be omitted: certain functions will only occur if their corresponding precursor functions have already occurred. For instance, if the hero does not enter into a chase, then they do not need to be rescued. This permits a relative degree of freedom within the confines of a defined structure, which is an ideal starting point for a computationally creative project.

Propp’s *Morphology* has previously been utilised by Pablo Gervas, whose work builds upon Propp’s theory in an effort to generate entirely new plot structures, and also employ self-evaluative rating systems (Gervas (2013), and Gervas (2016)). *Chronicle* differs in that it focuses on variance of content and the pursuit of unexpectedness within an existing morphology, and the ratings it assigns relate to user expectation rather than a score based on conformance with a structure.

Chronicle also utilises an additional level of detail in each of the story functions, resulting in generated outputs constituted of fully-fledged sentences and paragraphs (with substitutions for randomised lexis, where appropriate), and seeks to incorporate subtle elements of humour. Considering *Chronicle*'s output resembles a typical story (rather than a series of short sentences devoid of detail), it encourages a more genuine response from human volunteers, thereby overcoming Gervas' issue with evaluators having to interpret "abstract representations". This is also a problem experienced by Rafael Pérez y Pérez, despite the nuanced plot structures his system generates (Pérez y Pérez, 2015). A story without detail is like a skeleton with no muscle, or the foundations of a house with no walls; we can recognise that it is a solid framework, but it lacks significant value when deprived of its defining details.

Output Evaluation

It is relatively simple to analyse the output of a computer that deals with hard data, because we know that 1 is always bigger than 0, smaller than 2, and not equal to 574. It is comparatively much more difficult to analyse creative output, particularly in a manner which enables meaningful comparison between multiple outputs. This is because the quality of a creative material is largely subjective. Once it has been established that a creative output satisfies the rules of grammar and incorporates a sufficient vocabulary, it largely becomes a matter of opinion as to how good (or otherwise) the text may be.

Additionally, there are multiple interpretations of what constitutes a 'good' text. For instance, a pithy crime thriller may have the capacity to excite a reader, but could fall short in the realms of literary excellence. Similarly, a great work from the literary canon may be considered the pinnacle of narrative innovation, but could send a reader to sleep should they attempt to negotiate it.

As such, a standardised evaluative process should be employed in order to overcome these hurdles, either eliminating or sufficiently accounting for the subjectivity of human evaluators.

Human Evaluation Anna Jordanous outlined the *SPECS* methodology, which is a three step process (Jordanous, 2012). The evaluator must declare their definition of creativity (in regard to the system they are evaluating), identify the standards for which they will be testing, and then test the system using those standards. It prioritises targeted feedback on specific aspects rather than an arbitrary numeric "creativity score".

The use of targeted feedback is certainly appropriate for attempting to quantify subjectivity, but the resulting lack of hard data increases the difficulty of accurate evaluation and comparison. Jordanous recognises this, and discusses the complications of attempting to reach a consensus while handling differing opinions.

The best that can be done when employing human evaluation is to follow a strict set of principles as objectively and realistically as possible, but even then, issues are likely to be encountered.

Self-Evaluation Rafael Pérez y Pérez outlined the "three layers" evaluation model as a potential evaluative methodology (Pérez y Pérez, 2014). This differs from Jordanous' approach in that it eliminates human opinion from the evaluation process. The first "layer" ensures that the plot is suitable and valuable enough to be evaluated. The second "layer" evaluates the "core characteristics" of the plot, in this case "climax", "closure", and "novelty". The third "layer" is responsible for "enhancers and debasers" which modify the overall "score" either positively (for original value) or negatively (for repetition).

This evaluative model is able to translate a creative output into hard data, which is arguably much more desirable than comparing subjective human feedback since it allows for clearer comparison and analysis. However, any plots or stories requiring evaluation need to have followed a particular format for the model to be applied correctly.

We may also wish to consider the implications of a computer evaluating its own output: if the output necessitates evaluation in the first place, it is indicative of the fact that we doubt its creative capabilities; as such, is it right that we rely on the same system to evaluate the output? Conceptually, this may sound like a valid concern, but in practice (due to the algorithmic nature of the evaluative process) this does not appear to be an issue.

Pérez y Pérez utilises a self-evaluative process referred to as an "engagement-reflection" (E-R) model (Pérez y Pérez, 2015). "Engagement" constitutes the generation of sequences and events, while "reflection" is responsible for evaluation and modification of the generated material. The story writing process is passed back and forth between two "agents" (each with a differing set of characteristics) which engage the E-R cycle at every step.

Similarly to Pérez y Pérez's three-layered approach (2014), this process encourages evaluation on multiple levels for each step of the story generation process, creating an autonomous feedback loop. The addition of a second agent simulates a collaborative environment in which multiple 'writers' contribute to the story, which more closely portrays the human creative experience (whether the second 'writer' is actually another person, or merely representative of the first writer's awareness of context). Due to the usage of numeric values for "emotional links" and the visual representation of agent "contextual knowledge structures", the process provides an amount of hard data that can be utilised for objective comparison.

Self-evaluation is not always this effective in its implementation. Gervas' 2013 project results in the output of an "abstract representation", which is "plagued with difficulty" as far as human evaluators are concerned, and likely to lead to "difficulty of interpreting the representation". This means that for Gervas, a system based evaluation is the only option.

In principle, this level of self-evaluation may sound like a desirable autonomous feature. However, Gervas mentions these "quantitative procedures" with little detail. With no third parties able to interpret or quantify the output, and when the evaluation system runs "at a corresponding abstract level", the reliance of this self-evaluation is

questionable. His more recent research (Gérvás, 2016) suffers from the same evaluation issue: it operates at an “abstract level”, and cannot be evaluated (or assessed for accuracy) by human volunteers.

Self-evaluation can be an incredibly powerful tool, allowing multiple stories to be autonomously generated and evaluated in the time it would take a human evaluator to even begin to read a single story. However, it must be possible to test the quality and accuracy of the self-evaluation process in order to quantify its efficacy and reliability, and this likely requires one or many human evaluators. As such, a desirable approach would be to build a system that can be evaluated by both human and computer, and does not rely too heavily on one or the other.

Implementation

Chronicle was developed in Java. It establishes a number of features whose functions need to be closely studied in order to achieve a greater understanding of the system as a whole.

Propp Story Functions

A method was implemented for each of Vladímir Propp’s thirty-one story functions. The characters that appear within these functions each correspond to one of Propp’s seven archetypes, such as “hero”, “villain”, and “princess” (Propp, 1968). Method calls are structured in such a way that no function will ever occur out of sequence, thereby maintaining narrative consistency. The manner in which these methods are negotiated is determined by probability and the system’s decision making process.

Similarly to Pablo Gérvás’s approach, the generation process ends “when the end of the sequence is reached” (Gérvás, 2016). Depending on the events of the story, this can potentially result in a premature ending in which the hero does not succeed in completing their quest.

Decision Making & Probability

Decision Making Decisions made during the story generation process are determined by probabilities based on what the average reader is likely to expect. For example, when the hero enters conflict with the villain, the typical reader will expect the hero to succeed. Thus, the probability of the hero succeeding is set at 80%, and the corresponding probability of the hero failing is set at 20%.

The system utilises roulette wheel selection in its decision making process; two probabilities are requested in the form of integers that sum to a total of 100, and boundaries are established based upon these integers. Using the above example, the system would establish boundaries of 1-20 for the hero failing, and 21-100 for the hero succeeding. A number between 1 and 100 would be randomly generated, and the system would return the corresponding decision based on which set of boundaries the value fell between.

Successive story functions are also determined with this process. Using the above example, if the hero failed the encounter, the narrative would end with their death. This would result in the omission of any remaining story functions. If it was determined that the hero succeeded in the

encounter, a second decision would be made to determine whether the villain was killed outright, or merely injured and therefore able to flee. If the villain was killed, then the hero’s journey home would be uninterrupted. If the villain had fled, then they would be able to pursue the hero on his or her journey home, leading to an additional encounter in which the hero would be endangered a second time.

Predictability Rating Every time a decision is made, the probability of that option’s occurrence is recorded cumulatively, along with the total number of decisions that have been made within the current story. An overall predictability rating can then be calculated for each story by dividing the cumulative probability by the total number of decisions. A higher predictability rating indicates a high level of predictability, whereas a lower rating indicates unpredictability.

Predictability Target Every story has a user-customisable predictability target that it will attempt to meet with its overall predictability rating. In order to do this, the system utilises a probability modifier.

Probability Modifier The system can invert the decision making probabilities by subtracting them from a probability modifier and returning the absolute value. For example, with a default probability modifier value of 0, and a decision with 80% probability:

$$0 - 80 = 80$$

...hence, the probability is unchanged. When the odds are tipped, the probability modifier is adjusted to 100, and the calculation is as follows:

$$100 - 80 = 20$$

...hence, inverted. This approach works in any scenario in which there are two decisions whose probabilities sum to a total of 100.

The system will re-evaluate its predictability rating every time a decision is made, and will either maintain or invert the probabilities in order to pursue its target. With this functionality in place, the system is able to dynamically monitor the progress of the story, and consciously make changes during its generation.

It is important to note that while the probabilities are inverted, it is still the original (non-inverted) probability that is added to the cumulative total. This is because we want the rating to continue to reflect the typical user’s expectations.

Word & Name Randomisation

Randomisation is typically a function that should be avoided in computational creativity, since it replaces autonomy with luck or random chance. However, its usage is justifiable in instances where the effects of the randomisation do not have a significant impact on the events of the story. Word randomisation is akin to a human writer selecting a different word from a thesaurus, and as such it can be justified as a reasonable introduction of variety into an otherwise repetitive story segment.

The system contains a bank of names (separated by gender) and words (separated into categories such as

locations, objects, colours, positive adjectives, negative adjectives, etc). In predetermined places, the system can be asked to pick a word from one of these categories. This means that while the structure of a particular segment may be quite similar from story to story, there is potential for lexical variety.

Article Selection: ‘A’ vs ‘An’

Due to the irregularities and inconsistencies of the English language, it is difficult to predict whether a word chosen at random should be prefaced with ‘a’ or ‘an’. We can have instances of “an honest man”, or “a honeybee”, and “a university” or “an unidentified object”. The system negotiates this issue by comparing the first few characters of a given word to a number of predefined cases in sequential order until a match is found, and succeeds in assigning the correct article to a magnitude of different words.

Gender Pronoun Selection: ‘Him’ vs ‘Her’

Each of the character archetypes can be portrayed by both males and females. In order to fully support this and maintain consistency, the system dynamically selects the correct pronouns based on the associated character’s gender. For example, with a male character, one passage might read: “The man opened his eyes”. With a female character, the same passage would read: “The woman opened her eyes”.

Sample Output

A number of stories generated by *Chronicle* (as well as the application itself) can be found online at the following location: <https://github.com/toddpickering/chronicle>

Below is an excerpt from a predictable story, at the moment the hero learns of the villain’s wrongdoing:

“Arthur hurriedly dialled his voicemail, and listened to the first message. It was Jenny. Her daughter had been kidnapped. Arthur took one last look at the desert, then turned and headed back towards the forest as quickly as he could. Upon arrival, he asked Jenny what had happened.”

The same event in the narrative can read quite differently when the system pursues an unpredictable story:

“Daisy hurriedly dialled her voicemail, and listened to the first message. It was Evie. Her son had been kidnapped. Daisy didn’t much care for Evie or her son, so she decided not to help, and spent the day at the canyon.”

The system has a number of opportunities to end the narrative (in an expected manner or otherwise). Below is an example of a particularly unexpected story in which the main character chooses not to leave their home, resulting in a story only a single paragraph long:

“There was a man named Oliver. He spent most of his time at the swamp. Nearby, there lived a man named Steve. Oliver was feeling especially unadventurous, and decided not to leave the swamp. Steve thought this incredibly boring.”

An excerpt from the ending of a much longer tale can be found below. At this point in the narrative, the false hero has been ridiculed after attempting to take credit for the hero’s actions, and is now seeking revenge.

“Darren was ridiculed for his foolish claims. He was furious with Karl, deciding it was all his fault. Darren wanted revenge.

Darren climbed to the top of the tree beside Karl’s house, intending to attack him when he walked by. Unfortunately for Darren, he was at the wrong house. After several hours, Darren began to tire. He lost his grip and fell from the tree, breaking his neck.”

User Test

Human-based evaluation can present issues of subjectivity. However, concepts such as creativity, originality, and unexpectedness are largely subjective themselves, and therefore can be difficult to evaluate from an analytical standpoint. These concepts play a crucial part in the understanding of how successful (or otherwise) *Chronicle* is. As such, it was necessary to acquire responses from third-party human participants in order to evaluate the supposed effectiveness of the software.

User Test Process

A survey was created and distributed online, containing a download link for the software and instructions on how to use it. The user was prompted to generate a story and read it in its entirety, and then assign the story two ratings.

The first rating concerned how entertaining the user found the story. The recording of this rating allowed comparisons to be made between user enjoyment, the user’s perceived predictability of the story, and the system’s perceived predictability of the story. ‘Entertaining’ is an open-ended, subjective word, and it was deliberately chosen for this reason. Since enjoyment is a subjective concept, it should be evaluated as such. Different people garner enjoyment and entertainment through different means. What is important is whether or not the user appreciates the creative output; their exact reasoning for this or the manner in which they do this is of less importance.

The second rating concerned how predictable the user found the events of the story. It was important to establish the scope of this question in an effort to avoid any existing impressions or biases that may be carried over from previous story generations. Ideally, we desired a higher predictability rating for stories that the software deemed to be predictable, and we desired a lower predictability rating for stories that the software deemed to be unpredictable. If this trend was followed, it would suggest that the software is accurate at identifying a user’s perceived predictability of a story.

This process was repeated until each user had read and evaluated six stories. Unbeknownst to the user, the first three stories had a predictability target of 100% (very predictable), while the latter three stories had a predictability target of 0% (not at all predictable). The goal was to have the users’ ratings match up with these targets.

It was important that the stories aiming to be unpredictable were generated last; generating these stories first could lead to inaccurate results, since a new user might evaluate a story as being unpredictable simply because they are unfamiliar with the software and have not yet picked up on its patterns, rather than the story being genuinely unpredictable as a result of the software’s deliberate intervention. However, there is a possibility of introducing ordering bias by following this strategy.

The first three (predictable) stories gave the user the opportunity to get acquainted with the software, during which they would likely have begun to pick up on some of its patterns. Then, once it came to generating the latter three (unpredictable) stories, the user would have been better equipped to evaluate the overall predictability.

After completing the testing process, the user was prompted to submit statistics generated by the system during their session, and was given the chance to submit written feedback about their experience with the software.

There are numerous reasons why users were asked to generate six stories. Firstly, it had to be an even number, so that there could be an even distribution of predictability targets (e.g. three stories with a 100% target, and three stories with a 0% target). Secondly, a smaller number of stories (such as two or four) would not have given the user adequate experience with the software. With so few stories generated, it would have been difficult for the user to identify any recurring patterns, potentially leading them to assign an inaccurately generous predictability rating. Thirdly, it was necessary to have a relatively small workload for each user to complete in an effort to retain user focus, and to encourage more participants to respond to the survey. In a project such as this (where subjectivity and opinion play significant roles) it is far more useful to have a wider range of users evaluating a smaller number of stories than it is to have a very small number of users evaluating many stories.

Results & Analysis

Twelve surveys were completed, each containing evaluations of six stories. This resulted in a total of 72 evaluated stories. User predictability and entertainment scores were rated on a scale of 1-5, then translated to a scale of 0-100 for clearer comparison with system ratings.

Each user generated their own unique set of stories so that their evaluation could include their direct interaction with the software. However, it is possible that this approach introduced noise into the evaluation.

The relatively small sample size and potential for noise is indicative of the fact that these are merely preliminary results, and as such correlation measures have not been included. More conclusive results and comprehensive statistical analysis could be pursued as further research.

Predictability: User Rating vs System Rating

Human and system predictability ratings were closely related for some users, but for other users the system's ratings appeared to be less accurate. The averaged ratings for each of the evaluated stories are shown in figure 1.

The data in figure 1 demonstrates an increase in user perceived predictability across the first three stories, which reflects the expectation that users would begin to notice patterns in the generated stories as they became more familiar with the software. Nevertheless, the system was successful in generating unpredictable stories, evidenced by the decrease in user perceived predictability for stories 4-5. The increase in predictability for story 6 suggests that by this point in the test, the users may have become accustomed to the system's typical output.

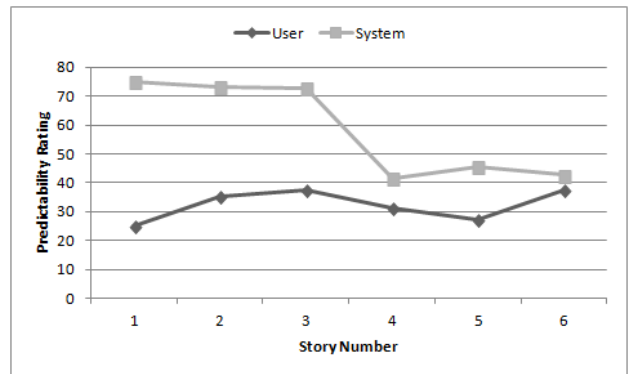


Figure 1: Averaged system and user predictability ratings for all 12 users. The lighter points indicate the ratings assigned by the system. The darker points indicate the ratings perceived by the user. A higher rating suggests a more predictable story, while a lower rating suggests a less predictable story. The system ratings have a mean of 58.5 and a standard deviation of 16.7. The user ratings have a mean of 32.3 and a standard deviation of 5.4.

Predictability: System Rating vs System Target

Figure 2 shows that while the system's predictability ratings never meet its targets precisely, it is capable of pursuing its assigned predictability target.

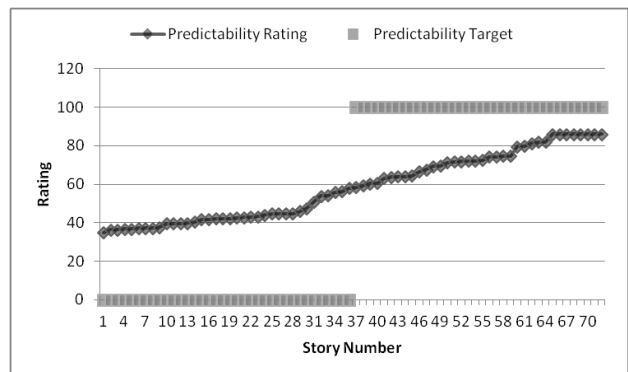


Figure 2: The system's predictability rating compared to the system's predictability target for all 72 evaluated stories, sorted by target (ascending) and rating (ascending).

User Entertainment vs User Predictability

Figure 3 demonstrates that a user's enjoyment of a story is inversely proportionate to their perceived predictability of said story: the more unpredictable a story, the more the user will enjoy it. This suggests that unpredictability is a desirable facet of a story generation system, aligning with Grace and Maher's assertion that "unexpectedness is [...] a vital component of computational creativity evaluation" (Grace and Maher, 2014), and validating *Chronicle's* pursuit of unpredictability.

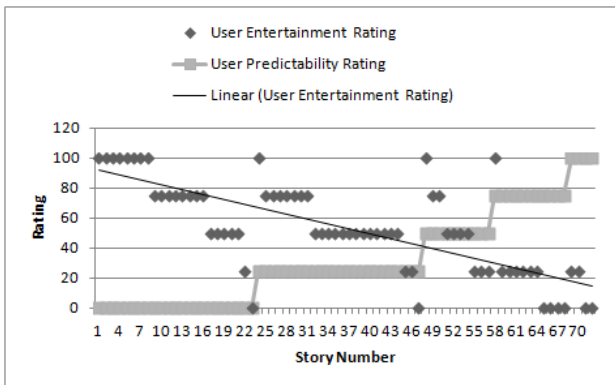


Figure 3: A comparison of user entertainment and predictability ratings for all 72 evaluated stories, sorted by predictability (ascending) and entertainment (descending).

User Entertainment vs System Predictability

Figure 4 demonstrates a similar (though admittedly much weaker) correlation between user entertainment and system predictability rating. This suggests that with a great deal more refinement, the system's predictability rating could potentially be used to predict a user's entertainment rating. However, in its current state, the system is quite a way from reaching such a goal.

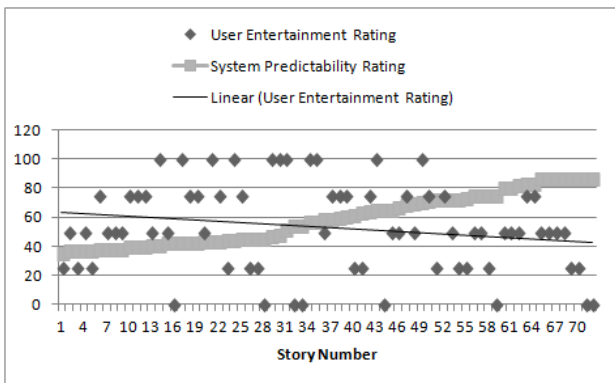


Figure 4: The users' entertainment ratings compared to the system's predictability ratings for all 72 evaluated stories, sorted by predictability rating (ascending) and entertainment rating (descending).

User Feedback

One user remarked that the endings which stood out the most and were the most enjoyable were those that were unexpected. Another user stated that it was the unexpected elements which made the greatest contribution to their enjoyment. A different user noted that on several occasions they were anticipating a certain end to the story, only for the system to develop an unexpected sub-plot.

Some users noted that it was difficult to keep track of the names of all of the story characters, and others

made similar remarks about location names. This can be attributed to the fact that generic names are used, and is likely exacerbated when reading six stories consecutively. Others noted that word selection was occasionally jarring or inaccurate, which can be attributed to the categorisation of vocabulary (despite already being separated by word type and positivity/negativity); just because a word is a positive adjective, it does not mean it can be used in all contexts. Another user stated that they felt a greater degree of variety could have been added to story events.

Evaluation

With a sample of only 12 users, there is not enough data to show statistically significant evidence. However, interesting results are suggested by the results and feedback obtained thus far.

Opinion and accuracy of results differed from user to user; such an outcome was inevitable in a project that, from a reductionist perspective, attempts to turn subjective opinion into hard data. Additionally, it is worth noting that a certain amount of unpredictability is always going to be sacrificed when following a predetermined structure, even with added variation.

The system appears to be fully competent in pursuing its assigned predictability target. Based on the test results, the predictability rating system shows promise, but lacks accuracy in relation to user ratings.

When reading the source code, it is simple to keep track of characters and locations, since they are all clearly defined by their field names. From the user's perspective, when randomisation is introduced, it becomes understandably more difficult. Memorable names (potentially based on status or character type) and locations would help to alleviate this issue. While it is worth noting that only a small number of users commented on this aspect, it is a clear example of the importance of considering the user's perspective at all times during software development.

The users' experience with the software may have been adversely affected by response bias, potentially introduced by the mention of 'predictability' in the survey questions. This is undesirable, but subtly attaining this rating without directly naming the concept would have been very difficult. Allowances could at least be made for this issue in the test mode of the software, in which statistics relating to predictability remained hidden from the user until completion of the process.

Many of the story functions would benefit from a reassessment of their level of detail; some contain too much, while others do not contain enough. This creates an imbalance in certain stories, especially if several functions of a similar level of detail are selected, potentially leading to a story whose detail is overwhelmingly prevalent, or decidedly minimal. However, a noticeable improvement (in regard to detail) has been made on existing projects such as Pérez y Pérez's *MEXICA* system (2015), in which stories are constituted of simplified one sentence segments which arguably struggle to encourage user engagement.

Some story functions could benefit with a more liberal usage of word randomisation to increase variety; the rigidity

of a predetermined structure needs to be offset by sufficient variation in content. However, refinement of the word randomisation process is necessary to ensure contextual consistency. This could be achieved through a more nuanced categorisation system for each of the vocabulary files, with each clearly defined by purpose and tone.

Chronicle succeeded in meeting each of the project objectives, although there is certainly room for improvement in order to improve both the system's accuracy and the efficacy of its features.

Further Research

Predictability Rating Refinement

Chronicle has demonstrated the potential of the predictability rating system, but also its inaccuracies. Refinement of this system could be approached by standardising probabilities based on actual user expectations in specific scenarios, as opposed to estimated expectations. This could be accomplished with a large scale user survey concerning the events of story scenarios, but would require a sizeable number of participants.

Alternative Narrative Theorists

Propp's *Morphology* has proven to be an effective foundation for many story generation systems. Future researchers might consider assimilating the work of different narrative theorists for the purposes of plot development or character design.

- **Joseph Campbell** outlined the 'hero's journey', in which seventeen 'stages' of a narrative adventure are divided between three 'acts': 'departure', 'initiation', and 'return' (Campbell, 1949). Similarly to Propp's *Morphology*, these 'stages' would be apt for assimilation into computerised functions.
- **Tzvetan Todorov** theorised that every story follows a structure involving the upset and re-establishment of the equilibrium (Todorov, 1971). This theory is comparable to Propp's *Morphology* in its usefulness: it defines a loose structure for a system to follow, but allows an even greater degree of freedom in regards to the events of each of these constituent parts.
- **Roland Barthes** proposed a theory that narratives are understood on the basis of five 'codes', each with a different function (Barthes, 1970). One example is 'enigma codes', which relates to a reader's necessity to unfurl mysteries or uncertainties as a plot progresses. These 'codes' could be deconstructed to formulate story functions.
- **Richard Bartle** established a relatively modern character theory model, ascribing four archetypes to humans playing video games in virtual worlds based on their intended goals (Bartle, 1996). This provides an ideal basis for systems that are driven by character actions and motivations, rather than predetermined structures.

Conclusion

Refinements can be made in a number of places, but *Chronicle* was successful in achieving each of the project objectives. There is a good amount of variance in the system's output. Propp's *Morphology* was successfully incorporated during the development process and utilised during story generation. The system is capable of modifying a story's narrative direction mid-generation, and does so based on its understanding of predictability. While the system's ratings are less accurate in some places, they show promise in others, demonstrating at least a partial understanding of predictability and unexpectedness.

Ultimately, *Chronicle* constitutes a valid contribution to the field, and presents a solid foundation upon which further research can be undertaken.

Acknowledgements

This work was undertaken by Todd Pickering in fulfilment of an MSc Computer Science project and dissertation, and was supervised by Anna Jordanous, whose support and guidance proved invaluable throughout.

References

- Barthes, R. 1970. *S/Z*. Paris: Editions du Seuil.
- Bartle, R. 1996. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research* 1(1):19.
- Campbell, J. 1949. *The Hero with a Thousand Faces*. New York: MJF Books.
- Gérvás, P. 2013. Propp's morphology of the folk tale as a grammar for generation. In Finlayson, M. A.; Fisseni, B.; Löwe, B.; and Meister, J. C., eds., *2013 Workshop on Computational Models of Narrative*, volume 32 of *OpenAccess Series in Informatics (OASISs)*, 106–122. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- Gérvás, P. 2016. Computational drafting of plot structures for russian folk tales. *Cognitive Computation* 8(2):187–203.
- Grace, K., and Maher, M. L. 2014. What to expect when you're expecting: The role of unexpectedness in computationally evaluating creativity. In Colton, S.; Ventura, D.; Lavrac, N.; and Cook, M., eds., *Proceedings of the Fifth International Conference on Computational Creativity*, 120–128. Ljubljana, Slovenia: Elsevier Procedia.
- Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4(3):246–279.
- Pérez y Pérez, R. 2014. The three layers evaluation model for computer-generated plots. In Colton, S.; Ventura, D.; Lavrac, N.; and Cook, M., eds., *Proceedings of the Fifth International Conference on Computational Creativity*, 220–229. Ljubljana, Slovenia: Elsevier Procedia.
- Pérez y Pérez, R. 2015. A computer-based model for collaborative narrative generation. *Cognitive Systems Research* 3637:30–48.
- Propp, V. 1968. *Excerpts from Morphology of the Folk Tale (American Folklore Society Bibliographical and Special Series)*. Indiana University Research Center in Anthropology, Folklore, and Linguistics, 2nd edition.
- Todorov, T. 1971. The 2 principles of narrative. *Diacritics* 1(1):37–44.

A Model of Inter-musician Communication for Artificial Musical Intelligence

Oscar Puerto & David Thue

Center for Analysis and Design of Intelligent Agents
School of Computer Science
Reykjavik University
Menntavegur 1, 101 Reykjavik, Iceland
oscar15@ru.is, davidthue@ru.is

Abstract

Artificial Musical Intelligence is a subject that spans a broad array of disciplines related to human cognition, social interaction, cultural understanding, and music generation. Although significant progress has been made on particular areas within this subject, the combination of these areas remains largely unexplored. In this paper, we propose an architecture that facilitates the integration of prior work on Artificial Intelligence and music, with a focus on enabling computational creativity. Specifically, our architecture represents the verbal and non-verbal communication used by human musicians using a novel multi-agent interaction model, inspired by the interactions that a jazz quartet exhibits when it performs. In addition to supporting direct communication between autonomous musicians, our architecture presents a useful step toward integrating the different subareas of Artificial Musical Intelligence.

Introduction

Artificial Musical Intelligence is a broad area of research that uses Artificial Intelligence (AI) techniques to build autonomous, interactive, musical systems (Collins 2006). Hiller (1959) was one of the pioneers of combining AI and music, building an application that generated musical compositions based on rule systems and Markov Chains. Later, Cope's (1992) "Experiments in Music Intelligence" used symbolic AI techniques such as grammars to generate musical compositions. Readers interested in the history of AI and music are encouraged to read Miranda's (2013) survey. We believe that the intersection of AI and music is an ideal context for the study of computational creativity.

Computational Creativity is the use of autonomous systems to generate and apply new ideas that would be considered creative in different disciplines of social life, including art, sciences, and engineering (Besold et al. 2015). In this paper, we focus particularly on the creativity that is inherent in collaborative, improvised, musical performance, and we adopt Roads's (1985) assertion that both composition and performance can be usefully tackled using AI techniques. Research and practical efforts that pursue these objectives are commonly discussed under the title "Musical Metacreation" (MuMe) (Eigenfeldt and Bown 2012), which can be well thought of as a subfield of Computational Creativity.

Musical Metacreation

The study of MuMe techniques has been approached from several different perspectives, which we roughly organize into three subareas: Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems.

Algorithmic Composition is a subarea of MuMe that seeks to automate different aspects of musical composition, including orchestration, score editing, and sound synthesis (Fernández and Vico 2013). For example, STELLA (Taube 1993) is an automated music representation system that can be used to edit musical scores.

Live Algorithms seeks to build autonomous systems that can perform in a collaborative musical setting, sharing the same privileges, roles, and abilities as a human performer. For example, IXI LANG (Magnusson 2011b) is an interpreted programming language that produces musical events in response to instructions typed in real-time – a practice known as "coding live music" (Magnusson 2011a) or simply *live coding*. Autonomy is a central concern when creating such systems, meaning that the interaction between the musician and the system must be strictly collaborative; neither should control the other. Elements of Algorithmic Composition, live electronics (e.g., IMAGINARY LANDSCAPE (Cage 1960)), and free improvisation are often combined to satisfy this constraint (Blackwell 2009). While Algorithmic Composition aims to model different elements of the composition task, Live Algorithms seeks to model the creative abilities that can be observed when human musicians perform.

Musical Multi-agent Systems is a subarea of MuMe that seeks to model the composition and performance of music as a task that requires collaboration between multiple agents. The general concepts of multi-agent systems can be applied to MuMe in two ways: multiple agents can be used to represent a single autonomous musician (e.g., a composer and a performer) (Murray-Rust, Smaill, and Edwards 2006), or the behaviour of multiple autonomous musicians can be represented as a single multi-agent system (e.g., a string quartet) (Wulfhorst, Nakayama, and Vicari 2003; Carôt, Krämer, and Schuller 2006; Thomaz and Queiroz 2009). Ideas related to computer networking are often used in this context, with communication protocols being defined and used to deliver messages between interacting agents.

Although many useful advances have been made in each of these three subareas, methods and architectures for com-

binning such advances remain largely unexplored. Recently, Bown, Carey, and Eigenfeldt (2015) developed “Musebot Ensemble” – a platform for agent-based music creation in which musical agents are developed individually and designed to accomplish a specific role in a larger ensemble. In their paper, the authors asserted that the agents in the ensemble (called “Musebots”) must be able to communicate musical ideas through a set of predefined messages, toward supporting collective performances that are analogous to those of human musical ensembles.

While the Musebot Ensemble platform offers a basis for integrating various MuMe techniques, its model of how agents communicate can be improved. Specifically, it lacks support for direct communication between agents, choosing instead to allow only a special, centralized “Conductor” agent to communicate directly with each Musebot. The result is that some agents can be left unaware of decisions that are made by other agents, reducing their ability to perform well (Eigenfeldt, Bown, and Carey 2015). Furthermore, direct communication between agents is essential to certain kinds of music, where the need for real-time coordination is inherent to the musical style (e.g., small-group jazz (Bastien and Hostager 1988)).

In this paper, we propose an architecture for Musical Metacreation that offers two contributions. First, it extends the Musebot Ensemble platform with a model of direct communication between autonomous musicians. Second, it does so in a way that facilitates integrating recent advances in the MuMe subareas of Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems.

The remainder of this paper is organized as follows. We begin with a brief formulation of our challenge and follow with an overview of related work, covering each of MuMe’s subareas in turn. We then present our architecture in two parts; we describe its overall structure and each of its component parts, and then explain how the parts interact with one another. We conclude by discussing our contributions and offering some suggestions for future work.

Problem Formulation

The study of human cognition and how it can be modelled has contributed to several improvements in our daily lives, such as “smart systems” that use AI techniques to ease the experiences of their users. Inspired by this perspective, we view the study and emulation of human musical abilities as an important avenue to explore in the pursuit of autonomous musical systems. As we described in the Introduction, one set of abilities that merits emulation are those that enable and support direct communication between musicians, spanning both verbal and non-verbal modes. Specifically, one’s abilities to negotiate, synchronize, compose, and perform with others are essential in the context of collaborative musical improvisation (Walker 1997). Furthermore, from their case-study observation of a jazz quartet’s performance, Bastien and Hostager (1988) concluded that such musicians engage in direct verbal and non-verbal communication across three distinct modes: instruction, cooperation, and collaboration.

In this work, we seek to extend the Musebot Ensemble platform in a way that facilitates direct communication be-

tween autonomous musicians, while at the same time supporting and demonstrating the integration of different techniques from the three subareas of MuMe.

Related Work

The community of researchers studying MuMe has grown over the years, with projects like the Musebot Ensemble platform and research networks like Live Algorithms for Music (LAM) seeking to encourage interest and integration in the context of musical creativity. As a result, numerous papers have been published in the MuMe subareas of Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems, and we consider several of them in the subsections that follow. We will conclude our review of related research by summarizing recent efforts to facilitate and promote integration across MuMe’s subareas.

Algorithmic Composition

Algorithmic Composition (AC) is an area of research that has contributed to several technological advances in the music industry, as many tools have been created to help musicians automate their composition tasks. In this section, we focus only on algorithms for composition that involve the application of AI, and particularly on those that do not require any human intervention.

Generative grammars and Markov Models are some of the first methods of AI that were used in AC (Rader 1974; Roads and Wieneke 1979; Rueda, Assayag, and Dubnov 2006; Keller and Morrison 2007; Morris, Simon, and Basu 2008). Abdallah and Gold (2014) offer a detailed explanation of grammar based models and Markov models, as well as a comparison between them. Researchers were also interested in evolutionary methods, in which a subset of solutions are generated from an initial set and then evaluated using a fitness function to measure their quality (Coello et al. 2007). Another method that is widely implemented in AC is Artificial Neural Networks, in which interconnected processing units are typically used to accomplish a pattern recognition task (Yegnanarayana 2009). For example, Goldman et al. (1996) developed a hybrid system based on the communication and cooperation of agents. These agents applied heuristic rules to solve problems relevant to polyphonic music composition, in real time. The melodies produced by the system are generated by neural networks that predict the expected value of each subsequent note in the melody. Although Goldman et al. successfully modelled some aspects of inter-agent communication (such as agreeing on which notes to play together), other important aspects were not included in the model (e.g., cueing transitions or negotiating over necessary tasks). Furthermore, their system focused primarily on modelling the cognitive processes of a human musician and their individual capacity to undertake different musical tasks (e.g., analyzing possible combination of notes and performing them at the same time). Nishijima and Watanabe (1993) used ANNs to learn musical styles. An overview and taxonomy of methods in AC is provided in Fernández and Vico’s (2013) survey.

Despite the capacity of AC to automate certain aspects of music composition, it remains challenging to represent features that are exclusively in the domain of communication between agents, such as the ability to share musical ideas with another musician.

Live Algorithms

Much research on the topic of Live Algorithms has focused on the challenges of sound analysis and beat tracking, toward allowing autonomous musicians to synchronize their performance and effectively take turns with human musicians. An example of such work is ANTESCOFO (Cont 2008), an anticipatory score-following system based on the collaboration between two agents (an audio agent and a tempo agent). These agents allow the system to synchronize accurately with its musical partner. An interesting feature of this system is its capability to predict changes to the structure of the music and follow those changes precisely in real time, providing an atmosphere of accompaniment with its partner. Similarly to ANTESCOFO, GENJAM is capable of performing alongside a human musician. GENJAM (Biles 2002) is a jazz improvisation system that evolves musical ideas trained by a human mentor while playing them interactively with a human performer. Biles stressed that one of the most valuable features of his system is its ability to “trade fours or eights” – a part of jazz performance where soloists take turns improvising and exchanging musical ideas in a way that mimics human verbal conversation. Blackwell (2003) suggested that the interaction between musicians in a musical ensemble could be represented by the self-organization components of swarms. Alternatively, Harrald (2007) discussed interactive improvisation in musical ensembles using the game-theoretic concept of the Prisoner’s Dilemma. Finally, members of the research network “Live Algorithms for Music” have provided an extensive description about of Algorithms, where they classified the different attributes that a live algorithm must have (Blackwell, Bown, and Young 2012).

While we believe that the study of Live Algorithms is essential to the development of Artificial Musical Intelligence, it has thus far only addressed the challenges of building autonomous systems that can perform together with humans. In contrast, our work seeks to understand and address the challenge of having multiple autonomous systems perform together, without any reliance on human participation.

Musical Multi-agent Systems

While the common practice of collaborative music performance is represented by musicians playing together in the same physical space, Carôt, Krämer, and Schuller (2006) provided a different perspective. In their paper, they discussed the challenges of *network music performance*, where musicians perform collaboratively from separate physical places, using the Internet as a communication channel. The notion of a network music performance provides a compelling abstraction for the study of Musical Multi-agent Systems, since interaction over a network requires a formalization of communication protocols that provide rules to govern the exchanges of messages between agents. For example,

Wulfhorst, Nakayama, and Vicari (2003) described a way to implement multi-agent musical interaction while also describing the representation of cognitive musical agents. Recently, various protocols have been designed by researchers in Multi-agent Systems. Murray-Rust, Smaill, and Edwards (2006) described an example of a protocol based on speech acts, where two agents (a musician agent and a composer agent) use a formal set of musical acts to establish an accurate understanding between them. INMAMUSYS (Delgado, Fajardo, and Molina-Solana 2009) is a multi-agent system that aims to create music in response to a user-specified profile of emotional content. The system composes a piece of music that attempts to meet the given emotional profile (e.g., users can request a “happy” song).

While several efforts to create autonomous music systems have used a multi-agent approach, the majority of them have modelled *each autonomous musician* as a multi-agent system, while saying little about how a group of such musicians should coordinate or interact. Furthermore, the integration of a multi-agent system framework in the context of the Musebot Ensemble platform (i.e. representing a Musebot as a multi-agent system) remains unexplored, and we believe that such work could benefit the interaction between agents in the Musebot Ensemble platform.

Efforts Toward Integration

Workshops on MuMe are held annually in conjunction with the International Conference on Computational Creativity (ICCC), toward inspiring collaboration and integration between artistic and technological approaches. The Musebot Ensemble platform arose as a result of this effort, with the particular goal of promoting both collaboration and the evaluation of work done in the field (Bown, Carey, and Eigenfeldt 2015). Eigenfeldt, Bown, and Carey (2015) presented the first application of this platform, including specifications for their Musebots, the architecture of the Musebot Ensemble platform, and a discussion of its benefits and limitations. They further asserted that the platform does not take precedents from a human band, but in our view, the use of such precedents holds great potential for advancing our knowledge of musically creative systems (as we argued in the Introduction). Finally, Thomaz and Queiroz (2009) developed a framework that aims to integrate several ideas from previous work (including pulse detection, instrument simulation, and automatic accompaniment) in the context of Musical Multi-agent Systems. While this framework offers a convenient layer of supporting functionality (eg., synchronization), it does not support the kinds of direct communication between agents that we pursue in this work.

General Architecture

Our goal is to extend the Musebot Ensemble platform in a way that supports direct communication between Musebots while simultaneously offering a clear avenue for integrating recent advancements in MuMe’s subareas. We have chosen to use the concept of a jazz quartet as a case study for this work, since jazz is a genre of music that routinely requires real-time coordination and improvisation from its players.

It thus serves as a suitable and convenient proving ground for the techniques of Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems. Furthermore, jazz performance (among humans) has been studied from the perspective of social science due to its inherent social interactivity, providing us with a solid point of reference when considering whether and how autonomous musicians can be made to play jazz. We will base our discussion on the work of Bastien and Hostager (1988), who presented a study of how four jazz musicians could coordinate their musical ideas without the benefit of rehearsal and without the use of sheet music. In their study, the authors found that the musicians communicated through a variety of different means, including visual, verbal, and non-verbal cues. We aim to model such communications between autonomous musical agents.

To extend the Musebot Ensemble platform in a way that supports direct communication between Musebots, we propose a two-level agent architecture in which, unlike previous work, each Musebot is itself comprised of multiple interacting agents. Figure 1 shows a graphical representation of our architecture, using part of Bastien and Hostager’s jazz quartet as an example (a saxophonist and a pianist).

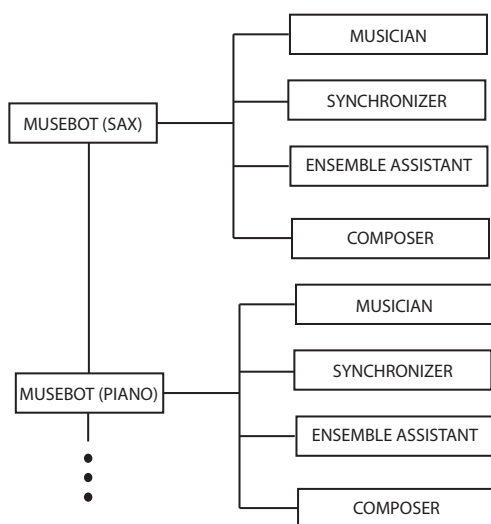


Figure 1: An example of our Musebot agent architecture. Each Musebot represents an autonomous musician and is a multi-agent system composed of four agents: a musician, a synchronizer, an ensemble assistant, and a composer.

At the top level (on the left of the figure), each Musebot represents a single autonomous musician such as a saxophonist or a pianist. At the lower level (on the right of the figure), each Musebot is made up of four different agents (musician, synchronizer, ensemble assistant, and composer) which are designed to distribute the various tasks that a Musebot should perform.

Agents at the lower level share a common goal: to help the Musebot perform appropriately as part of the Musebot Ensemble, including communicating and interacting effectively with other Musebots. To achieve this goal, a num-

ber of actions are executed by the agents based on the role that this Musebot has been given within the musical ensemble. For example, common roles in a jazz quartet are “leader/soloist” and “accompanist”, and their actions could include the leader requesting an accompanist to play an introduction for the next song. We describe each of the agents separately in the subsections that follow, and then explain how they interact thereafter.

We modeled our agents using finite state machines, similarly to Barbuceanu and Fox’s (1995) prior work in the context of industrial manufacturing. Barbuceanu and Fox modeled the communication between intelligent agents along a supply chain, using a framework based on a coordination language to represent different levels of coordination. This language allowed them to model a variety of interactive conversations using a variety of different finite state machines.

Musician Agent

In our architecture, the Musician agent is the primary component of a Musebot. It is responsible for carrying out the Musebot’s role in the ensemble (e.g., soloist or accompanist), and in doing so, it interacts with the rest of the agents in the architecture. The behaviour of this agent is represented by the finite state machine shown in Figure 2. After registering with a service that tracks the set of musicians in the current ensemble, the Musebot that is designated the leader (e.g., by an external user) will retrieve the structure of the song, which is set prior to the performance by an external user. Then, the leader will share the structure with the rest of the musicians. Next, it will negotiate the introduction to the song by requesting for another Musebot to agree to play an introduction. In case every Musebot refuses or fails to play the introduction, the leader will continue trying to find someone that wants to cooperate. From there, it will remain ready to improvise a solo whenever it feels appropriate. Finally, it will either request an ending to the song (similarly to how it requested an introduction) or it will pass the leadership to another musician, supporting the new soloist from then on as an accompanist. For Musebots that are initially designated as accompanists, registration is followed by receiving the structure of the song from the leader and either accepting or rejecting the leader’s request of play the introduction. From then on, it will play an accompaniment to the song (following the given structure) while waiting for a request to either end the song or become the next soloist.

Synchronizer Agent

One of the functions of the Synchronizer agent is to store information about events that happen during the progression of a song. For instance, when a Musebot’s musician agent is ready to perform an introduction, it will inform its synchronizer of the time at which it started to play, along with the expected duration of the introduction. This information will then be stored by the synchronizer and kept available for sharing with the rest of the agents through an interaction protocol. This mechanism allows any agents that ignore this information to request and calculate the time at which the introduction will be finished, toward knowing when they must

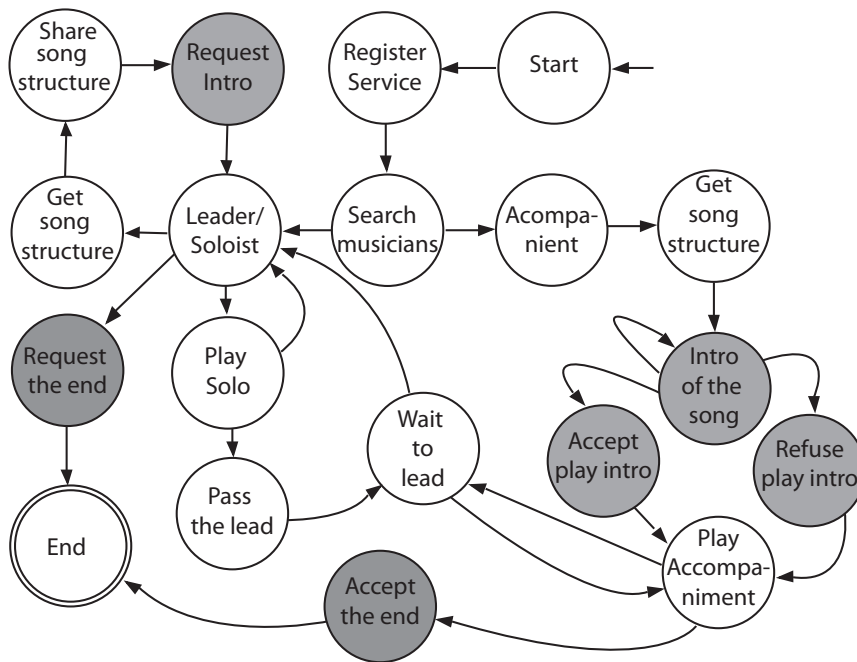


Figure 2: The Musician agent's finite state machine. Shaded states are involved in the example from Figure 3.

play their part of the song. This agent also provides a point of integration for recent advances in Live Algorithms.

Ensemble Assistant Agent

This agent serves as an intermediary between the Musebot Conductor (which is required by the Musebot Ensemble platform) and each Musebot. The Musebot Conductor provides a way for external users to control the ensemble (e.g., varying tempo, volume, or which Musebots are involved) (Eigenfeldt, Bown, and Carey 2015).

Composer Agent

The goal of this agent is to compose melodies, chord progressions, and/or solos at run-time. Each composition will depend on the role of the agent's "parent" Musebot and the instrument that it is playing (e.g., a string bassist would be less likely to play chords than a pianist). Coordinated with the help of the synchronizer, our implementation constructs each composition using JMusic, a Java library that encodes music as a symbolic representation analogous to CPN (Common Practice Notation) and plays it using the JAVA MIDI soundbank. This agent provides a point of integration for recent advances in Algorithmic Composition.

Interaction Between Agents

The interaction between agents in our architecture is managed by two different mechanisms: the Musebot Ensemble platform and a variety of interaction protocols. The interaction required by the Musebot Ensemble platform is defined by a specific set of messages (Eigenfeldt, Bown, and Carey

2015), which are exchanged between the Musebot Conductor and the Musebots. The messages are human-readable and classified into categories. For example, the message "/mc/time" is broadcasted by the Musebot Conductor to every Musebot in the ensemble, conveying the tempo of the composition for use in synchronizing the agents. Similarly, "agent/kill" is a message that indicates that the particular agent receiving this message should stop performing. In our architecture, this interaction mechanism is handled by the Ensemble Assistant agent; its principal task is to interpret these messages and transmit them to the multi-agent system.

Interaction Protocols

We developed our Musebots using JADE (Java Agent Development Framework) an agent-oriented framework designed in compliance with FIPA specifications. FIPA (Foundation for Intelligent Physical Agents) is an organization that provides an agent communication language along with standards for a number of interaction protocols (FIPA 2002). These interaction protocols are represented as sequences of messages (based on speech acts) for handling different actions between agents such as agreements, negotiation, and others (Bellifemine, Poggi, and Rimassa 1999). An example of a FIPA interaction protocol that we have implemented is shown in Figure 3.

We used the FIPA Contract Net Interaction Protocol to model part of the interaction between musicians that Bastien and Hostager (1988) described in their work. Specifically, prior to performing a song, the jazz quartet took some time to discuss which musician should play an introduction to the song. The Contract Net Interaction Protocol provides all of

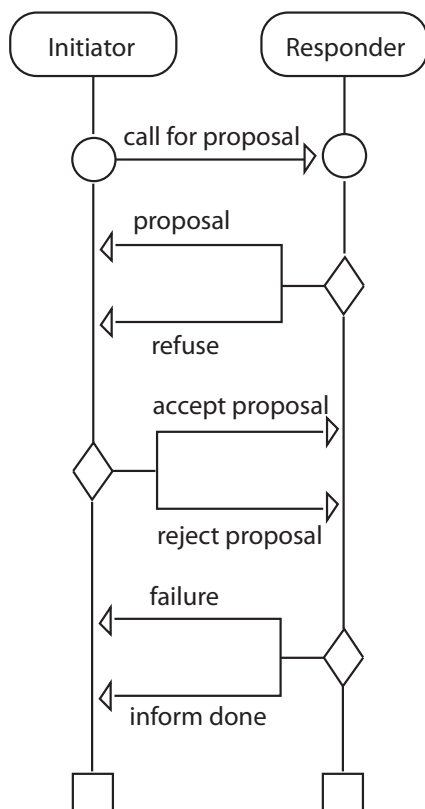


Figure 3: One of the agent interaction protocols that we implemented to support communication in our agent architecture (the FIPA Contract Net Interaction Protocol).

the necessary elements to represent this negotiation. The Musebot designated as the leader becomes the initiator of the conversation while the rest of the Musebots become responders. The initiator will send a call for proposal to the responders, proposing that one of them should play an introduction. Each responder will then reply with a proposal to play the introduction or a refusal to play it. The initiator will evaluate the received proposals, accepting one of them and rejecting the others. Once the proposal is accepted by the initiator, the responder will compose and play the introduction to the song and inform the initiator that the action was successfully completed. In case of failure by the responder, the initiator will repeat the conversation until the introduction gets played. While this example describes an interaction between agents at the top level of the architecture (e.g., a saxophonist interacting with a bassist, a drummer, and a pianist) other conversations are carried out internally by the agents at the lower level. For instance, there is a constant communication between the musician agent and the synchronizer agent each time a piece of the composition is planned to be played. Furthermore, the components of the different Musebots can also communicate one to another. One case where this occurs is between the synchronizer agents, which collectively share their information to

ensure that every Musebot's musician agent will be able to coordinate timings with the others. For example, the pianist might need to know when a section of the chorus (played by all musicians) will be finished, so that it can be ready to play a solo at that time.

Discussion

We have presented an architecture for Musical Metacreation (MuMe) that pursues the goal of integration across MuMe's subareas and extends the capabilities of the Musebot Ensemble platform. At the time of writing, our implementation of this architecture is well underway, with completion expected within the next two months.

Compared to previous approaches, our architecture offers certain benefits. By extending the Musebot Ensemble platform rather than attempting to replace it, it supports some cross-compatibility with different implementations of the platform. For example, given an ensemble made up of Musebots defined using our architecture, adding an arbitrary other Musebot into the ensemble (i.e., one which does not implement our architecture) should result in as viable a performance as the Musebot Ensemble platform allows. However, since the new Musebot's abilities to communicate with the others would be effectively reduced (because it lacks our architecture), the resulting ensemble performance might be impaired. Testing these hypotheses with a variety of different Musebots remains as future work.

The second benefit offered by our architecture is its ability to represent direct communication between Musebots using standardized protocols (e.g., the FIPA Contract Net Interaction Protocol). Eigenfeldt, Bown, and Carey (2015) claimed that there is no need for conversations between the agents in the Musebot Ensemble, since everything can be handled by passing messages through the Conductor. However, having conversations based on interactive protocols allows Musebots to negotiate, coordinate, and plan autonomously in a peer-to-peer fashion, which is a closer representation of how human musicians perform.

The third benefit of our architecture is that the multi-agent design of each Musebot offers convenient points of integration for recent advances in both Live Algorithms (in the Synchronizer Agent) and Algorithmic Composition (in the Composer Agent). In addition to the precedence offered by prior work, our choice to model each Musebot as a multi-agent system has some support from the field of Neuroscience. Specifically, Zatorre, Chen, and Penhune (2007) measured the activity of different areas of the brain during music performance, finding relationships between motor function and auditory interaction. While we do not claim that the specific agents in our architecture represent an optimal design or the true operation of the human brain, we have found that utility can be gained from having a multi-agent representation for each autonomous musician.

Finally, a fourth benefit of our architecture is that it allows Musebots to mimic two modes of communication that are used commonly when (human) musicians perform, Non-verbal cooperative modes (e.g., visual and musical cues) are mimicked when our Musebots attempt to pass the lead to another Musebot, and non-democratic instructive modes (e.g.,

sharing information about the next song) are mimicked when our Musebots share the structure of the song with the other Musebots in the ensemble.

Conclusions and Future Work

There is still much to accomplish in the pursuit of Artificial Musical Intelligence and the goals of Musical Metacreation. We view our architecture as a step toward this direction, since it offers both an extension to existing, related work and a convenient basis for integrating other recent advances. Future development of the architecture will involve implementing techniques from Algorithmic Composition in the Composer Agent and from Live Algorithms in the Synchronizer Agent, as well as an analysis of the communicative behaviours in the architecture. We plan to study these behaviours by analyzing detailed transcripts of the simulations performed by our Musebot Ensembles, to identify the musical and communicative events that happen during the progression of the song and compare them to similar analyses of human musical performances. It would also be interesting to apply our model of communication in a jazz quartet to different genres of music, both with and without additional Musebots that do not implement our architecture. Finally, we hope that our integrated, communication-focused approach will encourage and support further collaborative work in Musical Metacreation.

References

- Abdallah, S. A., and Gold, N. E. 2014. Comparing models of symbolic music using probabilistic grammars and probabilistic programming. In *Proceedings of the 40th International Computer Music Conference (ICMC—SMC—2014)*, 1524–1531.
- Barbuceanu, M., and Fox, M. S. 1995. Cool: A language for describing coordination in multi agent systems. In *ICMAS*, 17–24.
- Bastien, D. T., and Hostager, T. J. 1988. Jazz as a process of organizational innovation. *Communication Research* 15(5):582–602.
- Bellifemine, F.; Poggi, A.; and Rimassa, G. 1999. JADE—a FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, 33. London.
- Besold, T. R.; Schorlemmer, M.; Smail, A.; et al. 2015. *Computational creativity research: towards creative machines*. Springer.
- Biles, J. A. 2002. Genjam: Evolution of a jazz improviser. *Creative evolutionary systems* 168:2.
- Blackwell, T.; Bown, O.; and Young, M. 2012. Live algorithms: towards autonomous computer improvisers. In *Computers and Creativity*. Springer. 147–174.
- Blackwell, T. 2003. Swarm music: improvised music with multi-swarms. In *In Proceedings of the AISB '03 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 41–49. AISB.
- Blackwell, T. 2009. Live algorithms. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
- Bown, O.; Carey, B.; and Eigenfeldt, A. 2015. Manifesto for a musebot ensemble: A platform for live interactive performance between multiple autonomous musical agents. In *Proceedings of the International Symposium of Electronic Art*.
- Cage, J. 1960. *Imaginary landscape, no. 1: for records of constant and variable frequency, large Chinese cymbal and string piano*. Number 1. Henmar Press.
- Carôt, A.; Krämer, U.; and Schuller, G. 2006. Network music performance (nmp) in narrow band networks. In *Audio Engineering Society Convention 120*. Audio Engineering Society.
- Coello, C. A. C.; Lamont, G. B.; Van Veldhuizen, D. A.; et al. 2007. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer.
- Collins, N. M. 2006. *Towards autonomous agents for live computer music: Realtime machine listening and interactive music systems*. Ph.D. Dissertation, University of Cambridge.
- Cont, A. 2008. Antescofo: Anticipatory synchronization and control of interactive parameters in computer music. In *International Computer Music Conference (ICMC)*, 33–40.
- Cope, D. 1992. Computer modeling of musical intelligence in emi. *Computer Music Journal* 16(2):69–83.
- Delgado, M.; Fajardo, W.; and Molina-Solana, M. 2009. Innamusys: Intelligent multiagent music system. *Expert Systems with Applications* 36(3):4574–4580.
- Eigenfeldt, A., and Bown, O., eds. 2012. *Proceedings of the 1st International Workshop on Musical Metacreation (MUME 2012)*. AAAI Press.
- Eigenfeldt, A.; Bown, O.; and Carey, B. 2015. Collaborative composition with creative systems: Reflections on the first musebot ensemble. In *Proceedings of the Sixth International Conference on Computational Creativity*, 134. ICCO.
- Fernández, J. D., and Vico, F. 2013. Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research* 48:513–582.
- FIPA. 2002. FIPA ACL message structure specification. *Foundation for Intelligent Physical Agents*, <http://www.fipa.org/specs/fipa00061/SC00061G.html> (30.6. 2004).
- Goldman, C. V.; Gang, D.; Rosenschein, J. S.; and Lehmann, D. 1996. Netneg: A hybrid interactive architecture for composing polyphonic music in real time. In *Proceedings of the International Computer Music Conference*, 133–140. Michigan Publishing.
- Harrald, L. 2007. Collaborative music making with live algorithms. In *Australasian Computer Music Conference*, 59–65. Australasian Computer Music Association.
- Hiller, L. A. 1959. Computer music. *Scientific American* 201:109–121.
- Keller, R. M., and Morrison, D. R. 2007. A grammatical approach to automatic improvisation. In *Proceedings of the Fourth Sound and Music Conference*. SMC.
- Magnusson, T. 2011a. Algorithms as scores: Coding live music. *Leonardo Music Journal* 21:19–23.

- Magnusson, T. 2011b. *ixi lang: a supercollider parasite for live coding*. In *Proceedings of International Computer Music Conference*, 503–506. University of Huddersfield.
- Miranda, E. R. 2013. *Readings in music and artificial intelligence*, volume 20. Routledge.
- Morris, D.; Simon, I.; and Basu, S. 2008. Exposing parameters of a trained dynamic model for interactive music creation. In *AAAI*, 784–791.
- Murray-Rust, D.; Smaill, A.; and Edwards, M. 2006. Mama: An architecture for interactive musical agents. *Frontiers in Artificial Intelligence and Applications* 141:36.
- Nishijima, M., and Watanabe, K. 1993. Interactive music composer based on neural networks. *Fujitsu scientific and technical journal* 29(2):189–192.
- Rader, G. M. 1974. A method for composing simple traditional music by computer. *Communications of the ACM* 17(11):631–638.
- Roads, C., and Wieneke, P. 1979. Grammars as representations for music. *Computer Music Journal* 48–55.
- Roads, C. 1985. Research in music and artificial intelligence. *ACM Computing Surveys (CSUR)* 17(2):163–190.
- Rueda, C.; Assayag, G.; and Dubnov, S. 2006. A concurrent constraints factor oracle model for music improvisation. In *XXXII Conferencia Latinoamericana de Informática CLEI 2006*, 1–1.
- Seddon, F. A. 2005. Modes of communication during jazz improvisation. *British Journal of Music Education* 22(01):47–61.
- Taube, H. 1993. Stella: Persistent score representation and score editing in common music. *Computer Music Journal* 17(4):38–50.
- Thomaz, L. F., and Queiroz, M. 2009. A framework for musical multiagent systems. *Proceedings of the SMC* 1(2):10.
- Walker, W. F. 1997. A computer participant in musical improvisation. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, 123–130. ACM.
- Wulfhorst, R. D.; Nakayama, L.; and Vicari, R. M. 2003. A multiagent approach for musical interactive systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 584–591. ACM.
- Yegnanarayana, B. 2009. *Artificial neural networks*. PHI Learning Pvt. Ltd.
- Zatorre, R. J.; Chen, J. L.; and Penhune, V. B. 2007. When the brain plays music: auditory–motor interactions in music perception and production. *Nature reviews neuroscience* 8(7):547–558.

A Ballad of the Mexicas: Automated Lyrical Narrative Writing

Divya Singh
San Jose State University
San Jose, CA
divya.singh@sjsu.edu

Margareta Ackerman
San Jose State University
San Jose, CA
margareta.ackerman@sjsu.edu

Rafael Pérez y Pérez
Universidad Autónoma Metropolitana
Mexico City, Mexico
rperez@correo.cua.uam.mx

*But she fell in love with him
Girl when they feel the same
The princess was in love with the priest
Can't let go and it never goes out*

*She also abominated what he did
Be the things they said
The princess was shocked by the priest's actions
And though her heart cant take it all happens¹*

Abstract

Recently, computational systems began approaching challenges that were previously considered to lay exclusively in the human creative domain, such as the art of storytelling and lyrics writing. In this paper, we explore combining these two art forms through the automated creation of ballads. We introduce MABLE (MexicA's BaLLad machinE), based on the plot generation system, MEXICA. Integrating both cognitive and statistical models, MABLE is the first computational system to write narrative-based lyrics. User studies demonstrate MABLE's success at creating emotionally-engaging lyrics with coherent plot.

Introduction

The making of a meaningful and engaging plot extends beyond grammatical structure, requiring an understanding of the underlying context as well as complex interactions amongst agents. As such, it is perhaps unsurprising that the automated creation of coherent stories continues to pose a substantial challenge. This is evident in previous work on lyrics and poetry generation, where the selection of related and powerful words can lead to emotional engagement, yet without the integration of a coherent plot. On the other hand, lyrical works written by humans across diverse genres often center around a consistent story.

In this paper, we tackle the integration of automated storytelling with lyrics writing. As such, we concurrently address challenges encountered in both of these domains, aiming for emotionally engaging lyrics endowed with poetic structure, all the while telling a coherent story.

¹Excerpts from a ballad created by MABLE.

The art of storytelling through song is found across many different musical genres and styles, yet this tradition is perhaps best embodied in the ballad. Originating from medieval French dance songs and later characteristic of poetry and lyrics from the British Isles, the *ballad* came to be known today as a song narrating a story, typically centered around a romantic, sentimental scheme. As such, we have chosen to initiate our exploration into narrative-based lyrics through the automated creation of ballads, creating lyrics that tell stories of love.

Our efforts in automating the writing of ballads lead to a new approach to their creation, which, to the best of our knowledge, has not been previously utilized by either humans or machines. Our method starts off with a complete plot-line, capturing the main elements of the story. The next step is used to endow the story with rhyme and rhythm. Specifically, every other line in our lyrics is devoted to the progression of the narrative, while the remaining phrases aim to transform the narrative into a ballad.

For the first phase, our system MABLE relies on MEXICA (Pérez y Pérez and Sharples 2001), a plot generation machine based on the engagement-reflection model for creative writing. MEXICA produces novel, coherent plots of stories about the Mexicas, an indigenous people of what is today Mexico City.²

After generating a plot using MEXICA, the second phase utilizes a statistical model to expand the plot into lyrics. For each line in the plot, the statistical model is used to create a new, poetic phrase that, by rhyming with the original sentence and following its metric structure, leads to seamless integration of the narrative into the ballad. Specially, we utilize a second-order Markov model trained on a corpus of love songs. Then, we select amongst sentences generated using this model through a set of constraints that allow MEXICA's sentences to fit with the new phrases. We believe that the integration of MEXICA's cognitive-model with the statistical model used in the second phrase is one of the most interesting aspects of our approach, and is perhaps central

²The first part of this paper's title comes from the common practice of naming ballads in the form "A ballad of the ..."

to our system’s ability to convey narratives in a poetic form.

To evaluate MABLE’s artifacts, we compare them against poems and lyrics created by previous systems, as well as human-made lyrics. Results point to MABLE’s success in creating emotionally engaging lyrics with a coherent storyline.

We begin with a summary of related previous work and a discussion of MEXICA, which is integral to MABLE. Next, we proceed with an in-depth description of our model, following which we present our user study and discuss findings and implications. We also briefly explore the potential of utilization MABLE for the creation of musical pieces by utilizing ALYSIA (Ackerman and Loker 2017), a songwriting system that creates melodies for user-provided lyrics. The paper concludes with avenues of investigation for future work.

Previous work

A wide range of approaches to the creation of lyrics and poetry have been explored, ranging from templates-based to statistical methods. Tra-la lyrics 2.0 (Gonçalo Oliveira 2015) offers a template-based approach for lyrics generation, combining two previous systems, PoeTryMe and Tra-la-Lyrics. In the former system, user-provided keyword are fed into a sentence generator that utilizes templates and a semantic network to create sentences. These are then used to fill in a poem template. Tra-la-Lyrics creates lyrics based on melodies by constructing text where the stresses match the rhythm of the melody. Tra-la lyrics 2.0 combines these two systems by integrating rhythm information from the melodies into PoeTryMe’s architecture.

Full-face poetry generation (Colton, Goodwin, and Veale 2012) constructs poems by extracting key phrases from newspaper articles, which are then combined with database of similes to create template-based sentences. These sentences are in turn combined on the basis of considerations such as their rhyming patterns, word similarity, and sentiment. Other work that creates poetry based on user provided text includes (Tobing and Manurung 2015) and (Misztal and Indurkha 2014).

SMUG (scientific music generator) (Scirea et al. 2015) explores a related direction by creating songs based on academic papers. The idea is to extract key words from the paper and to utilize them by filling one of several pre-defined song structures, which are subsequently embellished with elements such as onomatopoeias.

Systems have also been designed particularly for the creation of rap lyrics, which have a distinct style. (Hieu Nguyen 2009) relies on a data set of hip-hop lyrics, separated into choruses and verses. These corpus are used to construct two quadgram models, from which sentences are subsequently generated. Finally, the sentences are combined based on common themes and matching number of syllables in rhyming words. Another system for Rap lyrics, DopeLearning

by (Malmi et al. 2015), combines sentences from existing human-made lyrics. DopeLearning’s lyrics string together phrases selected from a data base of rap lyrics from 104 different artists, relying on a deep neural net and RankSVM to select each subsequent line on the basis of considerations such as semantic similarity and rhyming.

Other related work considers a co-creative approach to lyrics writing. For example, LyriSys (Watanabe et al. 2017) uses a topic transition model, and allows the user to select topics and specify parameters such as the number of lines and syllables per line. The user can then select and edit lines from amongst those suggested by the system.

Closely related to lyrics generation is work on automated poetry creation. For example, (Toivanen et al. 2013) use constraint programming with standard constraint solvers. A static library of constraints is utilized, such as syllables, number of lines, and rhymes. (Ghazvininejad et al. 2016) create topical poetry starting from a user-specified topic word, which is then used to identify 1000 similar words. These words are subsequently grouped in rhyming classes, and rhyming pairs are selected. Next, a final state acceptor is created, where each poem that it could generate satisfies sonnet constraints and the pairs of rhyming words are used to create a rhyming pattern. The paths are then extracted using a Recurrent Neural Network.

Despite extensive work in this area, none of the previous systems aim to automatically create lyrics (or poems) that convey coherent stories, which is our focus here.

MEXICA

MEXICA (Pérez y Pérez and Sharples 2001) (Pérez y Pérez 2015) (Pérez y Pérez 2007) is a system that characterizes a model of creativity in writing that develops short stories about the old inhabitant of what today is México City. It is inspired by the engagement-reflection account of writing as creative design (Sharples 1999).

In MEXICA, a story is defined as a sequence of actions. Each action has an associated set of preconditions and post conditions in terms of emotional links and tensions between characters. For instance, the precondition of the action “the hunter murdered the jaguar knight” might be the emotional link “hunter hates the knight”; the post condition of the action “the princess awarded the eagle knight” might be the emotional link “the knight is very grateful towards the princess”.

In MEXICA tensions represent conflicts, e.g. when the life of a character is at risk, when the health of a character is at risk (i.e. when a character is hurt or ill), or when a character is imprisoned. Each time an action is performed, new conflicts may arise. The model also includes what we refer to as inferred tensions, i.e. tensions that are activated automatically when the system detects that: 1) two different characters are in love with a third one (tension due to love competition); 2) when a character has two opposite emotions towards another

one (tension due to clashing emotions); 3) and when a character hates another character and both are located in the same position (tension due to potential danger).

In this way, each time MEXICA generates a new story, the system produces a sequence of actions, and information about the emotional relations and conflicts between the characters in the tale. This information is used by MABLE to extract the sentiments in story.

Description of the model

Lyrics rely on structural elements to create motion, the sense of being pushed forward towards the climax. Pat Pattison, College of Music Professor and lyrics writing and poetry expert (Pattison 2009), describes how to write poetry using sense-bound imagery to enhance a song's emotional impact on its listeners. He tells how a lyric structure can create motion, which, in turn, creates emotion. The main elements used to achieve this effect are rhyme, rhythm, the number of lines, rhyming scheme and rhyme type.

We utilize some of Pattison's guidelines with the aim of creating lyrics that carry an emotional affect on our audience. MABLE begins with a story narrative, provided by the narrative system MEXICA. The rest of the system is broken down into three main modules, which together turn the storyline into lyrics. In particular, this is achieved by extending the original narrative sentence through the addition of a new figurative line of lyrics after each story line. The final ballad consists of alternating lines, where each narrative sentence is followed by a figurative one.

The following are the three main modules of MABLE. The *Sentence Evaluator* creates a set of sentences that fit with a given narrative line using parameters such as rhyme quality, number of syllables, rhyming scheme, and rhyme type. The second module, the *Sentiment Analyzer*, selects from amongst this set of sentences ones that match with the story's sentiments. Finally, the *Integrator* connects the lines and, if needed, changes the figurative sentences into third person narrative. An overview of MABLE's architecture is illustrated in Figure 1.

A second-order Markov model, created using Markovify (<https://github.com/jsvine/markovify>), has been trained on a corpus consisting of lyrics of 129 love songs by various artists, totally 4697 lines. Titles of top 100 lyrics are taken from an online music catalogue (<https://www.last.fm/>) and the remaining titles from a NME music blog (<http://www.nme.com/blogs>) (that contains songs voted by twitter followers). Lyrics to these titles are crawled from 70-80s rock and pop songs available on azlyrics.com. We then cleaned the lyrics corpus by removing punctuation, extra white spaces and repeated text, such as multiple instances of a chorus.

After receiving a sentence from MEXICA's story, a batch of phrases created using the Markov Model is fed to the Sentence Evaluator. The Markov Model is repeatedly called until a set of at least 50 high quality

candidate lines are created. The quality of lines is measured on the basis of rhyming score and the number of syllables in the lines.

After candidate figurative lines are attained, they are passed to the Sentiment Analyzer to get emotionally connected with the story. We use linguistic resources to generate the lyrics. Phonetics transcription are taken from the CMU Pronouncing Dictionary, which contains over 134,000 words and their pronunciations. A sentiment analyzer, Twinword API (<https://www.twinword.com/api/>) is used for extracting sentiments of phrases and categorizing them into positive, negative and neutral. We now discuss MABLE's components in greater detail.

1. **Sentence Evaluator:** This component finds candidate figurative lines that rhyme with a given narrative sentence. It uses the Markov model to generate a batch of 100 new sentences having no more than 60 characters. It then evaluates them based on rhyming quality and number of syllables. If not sufficiently many high quality lines are found, new batches are generated, until we have at least 50 high quality sentences.

To evaluate rhyming quality, we start with a pool of words that rhyme (to various degrees) with the last word in the story line. To this end, we use NLTK (<http://www.nltk.org/>) to create the pool of words which is then used to find words in the batch that rhyme with any word in the bag of words. More specifically, we use `nltk.tokenize` for sentence tokenization, `nltk.corpus.cmudict.entries()` to discover rhyming words, and `nltk.corpus.cmudict.dict()` to get the pronunciation of end words.

For example, if the input narrative sentence is *the priest was ambitious*, then the bag of rhyming words may include the words *auspicious*, *dishes*, *discus*, *judicious*, ..., *officious*, *riches*, *suspicious*, sorted by their rhyme score. Note that the rhyming words rhyme with the original to various degrees.

Then, we search our batch (created using the second order Markov model) for sentences ending with words that rhyme with any word in our bag of words. This leads to a greater number of acceptable options than had we required perfect rhyming with the word *ambitious*. Moreover, this flexibility allows our ballads to more closely resemble human-made songs, which often include imperfect rhymes.

In order to figure out whether two words rhyme, we tokenize the words. To this end, we use the Carnegie Mellon University Pronunciation Dictionary³ to get phonetic transcription of words. In case a word is not found in the dictionary, we take the last letter of the word. All pronunciations are considered,

³The dictionary consists of over 127069 words and their pronunciations. Out of these, 119400 are assigned a unique pronunciation, 6830 have two and 839 have three or more pronunciations.

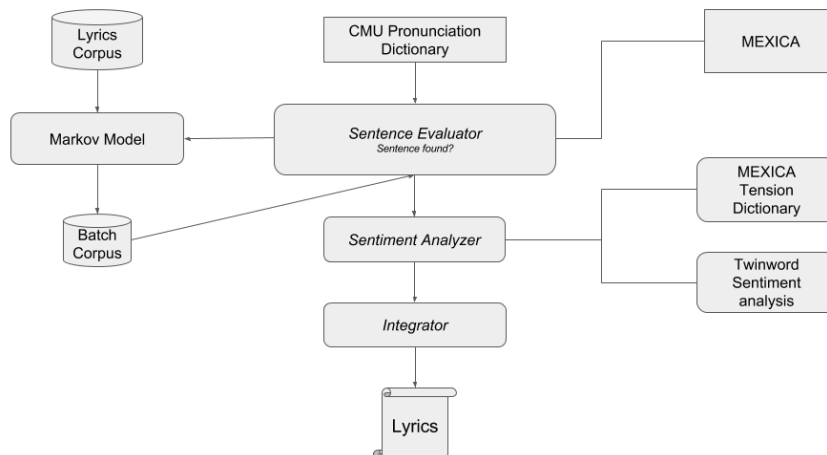


Figure 1: An overview of the lyrics generation workflow. The workflow depicted here creates a figurative sentence for MEXICA’s narrative line. The Sentence Evaluator repeatedly calls a Markov model to attain a list of candidate figurative sentences, which will follow the narrative line in the ballad. After a set of good candidate sentences is obtained, the Sentiment Analyzer finds the ones that express the same sentiments as those found in the narrative sentence. Finally, the Integrator adjusts the selected sentences so that they are told in third person narrative. The algorithm is repeated for each line in the narrative.

as the dictionary often provides multiple pronunciation options. For each of the pronunciations, we form a tuple list of candidate rhyming words from the CMUdict (<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>) lexicon and their rhyme qualities (how well they rhyme with the end word in the narrative line).

To calculate the quality of a rhyme, we take into account the degree of phonetic similarities such as end rhymes, slant rhymes, and assonance rhymes.⁴ We give high scores to words that end on the same vowel. An alternate rhyming scheme is used, which is a classic, frequently used rhyming scheme, also known as AABB.

We sort the candidate sentences based on their rhyme quality score, selecting the top 50 to go to the next step, which is to count the number of syllables.

CMUdict allows us to count the number of syllables in words, which lets us count the total number of syllables in each sentence. If no valid candidates are found, we return lines constructed using common interjections, while matching the number of syllables, such as “Oho oh oh oh oh.” This adds to the poetic feel of our lyrics.

⁴End rhyme is the most common type used in poetry, which occurs when the last syllables of words are matched. Its regular use marks off the ending of lines. It makes the lyrics sound appealing while making it easier for the audience to remember. Slant rhyme is formed by words with similar consonant sounds. Assonance rhymes do not have to rhyme in the traditional sense, but have matching vowels.

- Sentiment Analyzer:** Although following all the stylistic constraints of poems gives us well-structured results, we found the lines to be somewhat disconnected from the original story without additional processing. To further improve the lyrics, we incorporate sentiment analysis.

We connect the story lines with the newly generated ones by using the Twinword API (<https://www.twinword.com/api/>). This API returns a sentiment score for each word. The sentiment of each sentence is obtained by taking the average of its sentiment scores over all words. The score indicates how positive or negative is the overall line. If the score is below -0.05, it is tagged as negative, and anything above 0.05 is positive. Any line that falls within this range is neutral. We also utilize MEXICA’s tensions, such as *love competition* and *clash in emotions*. Whenever these tensions occur in a narrative sentence, MABLE selects candidate lines having opposite sentiments to that of the story line.

After sentiment analysis is done on the narrative sentence and the candidate figurative lines, we reduce our set of candidates lines to those that carry the same sentiment (positive, negative, or neutral) as the narrative line.

- Integrator:**

While MEXICA tells narratives in third person, most of the lyrics on which we train our Markov model are written in first person. As such, integrating the narrative lines with the figurative sentences often requires a change from a first to a third person point

of view.

The point of view decides the distance between the lyrics and the audience. It tells to what extent we know the storyteller and the actors in the story. So, what should we prefer, a first person narrative or a third person narrative?

In third person narrative, the singer acts as a storyteller who directs the audience's attention to an objective world that neither the singer nor the audience is a part of. It is similar to watching a movie while sitting in the audience (Pattison 2009). Neither we, the audience, nor the singer participate in the song's world. In this case, the singer acts as a storyteller. The third person narrative may be cleaner and more focused (Pattison 2009).

After we obtain the emotionally connected lines to our story, we neutralize the point of view using the Integrator. We substitute the first and second person pronouns with third person pronouns to connect the figurative lines with the narrative.

A Step-by-Step Example

To help illustrate our model, we present a step-by-step example creating a figurative sentence for a narrative line. Consider, for example, the following story line, given by MEXICA: "*The princess was in love with the priest.*" Our goal is to create a new sentence that will follow it in our lyrics. As such, we would like to create a sentence that rhymes with it, carries a similar sentiment, and has the same number of syllables. This process is subsequently repeated for each sentence in the narrative to form a complete ballad.

1. Sentence Evaluator

The sentence is tokenized into a list of words - ['The', 'princess', 'was', 'in', 'love', 'with', 'the', 'priest']. Using CMUdict, we attain the pronunciation of the last word. In this case, *priest* being the last word, we get [u'P', u'R', u'IY1', u'S', u'T']. Then, again using CMUdict, we get a bag of words that rhyme with *priest*: [beast, east, feast, increased, least...]. The numerical quality of rhyming words is calculated by the degree of phonetic transcription similarity of words. In particular, the rhyme quality is given by the number of consecutive matching pronunciation elements, starting from the end of the words. In case of ties, we give preference to words whose pronunciations end on the same vowels or words having the same vowels at the same places (for example, *moon* and *loop*.)

Then we repeatedly call *getMarkovBatch()* to get candidate sentences whose last words rhyme with the above bag of words, excluding sentences that end with exactly the same word.

Please note that the candidate lines given below do not perfectly rhyme with the word *priest*. In case perfect rhymes are not found, our model picks lower quality rhyming lines. Since imperfect rhymes are common in human-made lyrics, variety in rhyme quality also improves our lyrics.

Here are examples of some candidate sentences:

- Now that I have lost my light
- wondering where would I have got
- You can tell the lovers to part
- Down the highway of regret
- Mornings where blue and clouds of white
- So I stop and think about it

Next, we trim our list of candidate sentences to those whose number of syllables matches those in MEXICA's sentence. As such, our list is reduced to the following:

- You can tell the lovers to part
- wondering where would I have got
- The leader of the holy ghost
- Mornings where blue and clouds of white
- So I stop and think about it

2. Sentiment Analyzer

This step further reduces our set of candidates by performing sentiment analysis, and eliminating sentences that carry a different sentiment from MEXICA's sentence. Sentiment analysis rates each sentence into three categories: positive, negative, and neutral score range.

The following are some sentences from our list of candidate figurative lines along with their emotional score, where 1 represents positive, 0 represents neutral, and -1 corresponds to negative sentiment:

- You can tell the lovers to part, 1
- wondering where would I have got, 0
- The leader of the holy ghost, 1
- Mornings where blue and clouds of white, -1
- So I stop and think about it, 0

This gives us a final list of candidate sentences. Since the narrative sentence that we are working from has neutral sentiment, our final set of figurative sentence candidates consists of the following two:

- wondering where would I have got
- So I stop and think about it

Then, we pick a random sentence from the remaining figurative lines and pass it to the Integrator.

3. Integrator

After we select a figurative sentence, we pass it to the Integrator to substitute first person with third person pronouns.

Here are the resulting lyrics:

*The princess was in love with the priest
Wondering where would they have got*

The process is repeated for each of MEXICA's lines, giving complete lyrics. An example is shown in Figure 2. Every odd number sentence is created by

*The priest was born under grace of the great god
 And evolving from the shadows lifted
 The lady was an inhabitant of the great city
 But just remember there's a sign of intensity*

*The lady wanted him from the start
 The friends that they are
 The lady hid her love for the priest
 They just don't think they have got a secret*

*But she fell in love with him
 They don't know of the time
 The princess was in love with the priest
 Wondering where would they have got*

*The princess admired the lady
 Movies only make them crazy
 She felt much affection for her
 They thought the world were on fire*

*The priest was ambitious
 He will be out of place
 He wanted power easily
 As time goes by so slowly*

Figure 2: An example of a ballad made by MABLE.

MEXICA, whereas every even-numbered lyric sentence is created using the above process, by creating a suitable figurative line for each of MEXICA's narrative sentences.

Evaluation

Evaluation was performed through a user study comparing MABLE's lyrics with those created by two other systems, as well as lyrics written by humans. The subjects consisted of 30 students, 26 of which were undergraduate and 4 of which were graduate students, with an average age of 24 years old. Among these students, 26 were male and 4 were female. Subjects were not informed of how any of the lyrics had been made.

The survey asked them to rate lyrics along three parameters: coherence, emotional engagement, and the lyrics' overall quality. Specifically, the following three questions were asked for each of the lyrics provided: (1) Rate the storyline in these lyrics (is the plot coherent?) (2) How emotionally engaging are these lyrics (does it transmit feelings)? (3) Overall, how would you rate these lyrics?

Answers were provided via the standard 5 option Likert Scale, rating quality from *very poor* (1) to *very good* (5). For human-written lyrics, we selected professionally-written lyrics but which our subjects may not be familiar, as such avoiding the work of recently popular artists. Further, to align with MABLE's goal, we opted for lyrics that tell a story, choosing *Hall Of The Mountain King* by Denise Beadle, Peter Olliff, Peter Smith, and Edvard Grieg.

In addition to lyrics by MABLE, we have also included an artifact by a rap lyrics generator

(Hieu Nguyen 2009). To give a sense for the style, we include the first four lines here:

*my nasty new street slugger my heat seeks suckers
 now i'm a pimp you a player
 i'll rob boys ii men like i'm michael bivins
 if i'm from southside jamaica queens nigga ya'heard
 me*

Finally, we also compared with lyrics created with the full-face model (Colton, Goodwin, and Veale 2012). The first few lines of the lyrics we used are given here:

*the repetitive attention of some traditional african
 chants
 a heroic struggle, like the personality of a soldier
 an unbearable symbolic timing, like a scream
 blue overalls, each like a blueberry
 some presidential many selfless leaders*

Results are summarized in Figure 3.

Discussion

The development of a system that produces lyrics based on fables is an interesting challenge for computational creativity. One on hand it requires an agent capable of producing fables; on the other hand, it is necessary to modify such stories to satisfy lyrical requirements such as rhythm, rhyme, and so on. We are not aware of any other system that generates similar pieces.

Our results suggest that subjects clearly preferred the human lyrics. We believe that one of the reasons for it is the smooth integration of the entire text into a single coherent unit. MABLE takes a given story and then expands it, line by line, to generate lyrics. Our model not only adds rhetorical language that satisfies rhythm and rhyme constraints but which is also semantically related to the original phrase. The results presented in this work suggest that our approach is a good initial step, although much more work is required to integrate the new sentence in a more natural way.

Our study shows a correlation between the overall quality of a poem's and narrative's coherence. This suggests that subjects prefer lyrics with coherent narratives. As such, this may be an important characteristic for creative agents. This work is the first to tackle the challenge of developing lyrics with a coherent plotline, which may be essential for matching human quality lyrics.

Although MABLE was ranked highest amongst the computer-generated lyrics based on the coherence assessment, its advantage was reduced in the overall evaluation and it got second place in the emotional assessment. This result suggests that the use of strong language produced a clear impact in our subjects' evaluation of the lyrics. The poetry made by the rap lyrics generator (Hieu Nguyen 2009), which scored highest amongst computer generated systems on emotional engagement, contains words that evoke an emotional response (see above for an excerpt). Thus, we need to develop a mechanism that allows MABLE to analyse

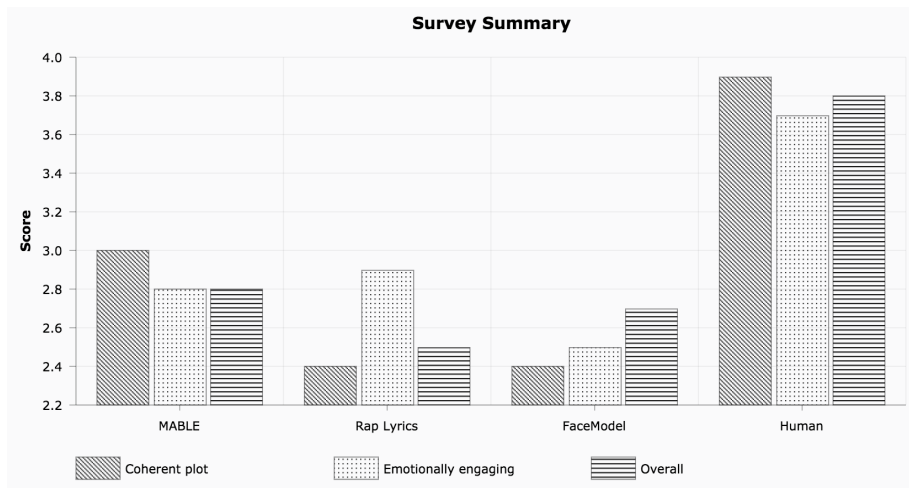


Figure 3: Average scores for each of the lyrics in our survey, along the following parameters: Coherent plot, Emotionally engaging, and Overall. While human-made lyrics clearly perform best on all criteria, MABLE’s lyrics score highest overall and on plot coherency amongst computer created poetry.

and, when appropriate, substitute some of the words in the figurative sentences with more emotionally impactful words. It would be interesting to explore whether genres outside of rap lend themselves to emotionally engaging word choices, and how such choices could be identified and utilized in an automated lyrics writer.

Setting it to Music

Before concluding, we briefly investigate the sonification of MABLE’s artifacts. Ultimately, lyrics are set to music, so that they could be expressed in their most natural form - through the singing voice. To explore the potential of MABLE’s lyrics to fit within musical works, we utilize ALYSIA (Ackerman and Loker 2017), a system designed to create melodies for user-provided lyrics.

Combining MABLE with ALYSIA brings us one step closer to autonomous songwriters. To examine the potential of their integration, we run ALYSIA on two verses created in a poem by MABLE. The sentences were given to ALYSIA one at a time, for each of which ALYSIA provided melody options (vocal melodies to which the lyrics could be sang). The melodies were subsequently selected and combined by Ai Nakamura, a computer science undergraduate student with an associates degree in music theory and composition. Figure 4 shows the resulting song.

Conclusions and Future work

Songs are a universal channel for expressing human needs, desires, and goals. Unfortunately, not everybody has the chance to pursue this form of self-expression. Our models can be used to support the development of skills required for producing lyrics. A first step might be to allow a collaboration between MABLE and human learners. For instance, MABLE would present the

user with examples of lyrics. Then, the system and the learner together would collaborate to generate new pieces; employing her routines for evaluation, MABLE could provide feedback to the user. Furthermore, a competition could be organized, allowing human judges to evaluate lyrics made by MABLE and the learners.

We believe that one of the most intriguing aspects of our work is the combination of different systems. Most notably, MABLE relies heavily on MEXICA’s narrative writing ability. Yet, we can go further. For instance, rather than generating a full story and then modifying it, we are analysing the possibility of integrating both processes. In this way, the unraveling of the plot might be constrained by the figurative language employed for the lyrics and vice versa. How does this integration affect the ER-Model and the production of figurative language? In the same way, we are interested in identifying parameters that allow for automatically producing (e.g. from the web) a corpus that better suits the content of the stories used for the generation of lyrics.

We also briefly explored interaction between MABLE and melody-writing system ALYSIA. Future work will explore in-depth collaboration between these systems. The aim of this collaboration is twofold. Firstly, we wish to create complete works, exploring their combined creative potential. Secondly, we would like to investigate modes of collaboration between these two system, exploring automated systems’ ability to collaborate on an artifact in a fully integrated fashion, perhaps resembling that of two human artists who may alter both the lyrics and melody in an iterative fashion.

References

[Ackerman and Loker 2017] Ackerman, M., and Loker, D. 2017. Algorithmic songwriting with alysia. *Inter-*

Princess' Minuet

Composer: ALYSIA
Lyricist: MABLE
Arranged: Ai Nakamura

$\text{♩} = 95$

But she fell in love with him Girl when they feel the same
mp

The pri-ness was in love with the priest Cant let go and it ne-ver goes out

She a - lso a - bo mi - na ted what he did Be the things they said
mf

The pri-ness was shocked by the priest's actions And though her heart cant take it all happens

Figure 4: Melodies for two verses created by MABLE. Each of MABLE's sentences were given to songwriting system ALYSIA, which created melodies for these sentences. The melodies were arranged by Ai Nakamura.

national Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART).

[Colton, Goodwin, and Veale 2012] Colton, S.; Goodwin, J.; and Veale, T. 2012. Full face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.

[Ghazvininejad et al. 2016] Ghazvininejad, M.; Shi, X.; Choi, Y.; and Knight, K. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1183–1191.

[Gonçalo Oliveira 2015] Gonçalo Oliveira, H. 2015. Tala-lyrics 2.0: Automatic generation of song lyrics on a semantic domain. *Journal of Artificial General Intelligence* 6(1):87–110.

[Hieu Nguyen 2009] Hieu Nguyen, B. 2009. Rap lyric generator.

[Malmi et al. 2015] Malmi, E.; Takala, P.; Toivonen, H.; Raiko, T.; and Gionis, A. 2015. Dopelearning: A computational approach to rap lyrics generation. *arXiv preprint arXiv:1505.04771*.

[Misztal and Indurkha 2014] Misztal, J., and Indurkha, B. 2014. Poetry generation system with an emotional personality. In *Proceedings of the 5th International Conference on Computational Creativity*.

[Pattison 2009] Pattison, P. 2009. *Writing better lyrics*. Writer's Digest Books.

[Pérez y Pérez and Sharples 2001] Pérez y Pérez, R., and Sharples, M. 2001. Mexica: A computer model of a cognitive account of creative writing. *Journal of Exper-*

imental & Theoretical Artificial Intelligence 13(2):119–139.

[Pérez y Pérez 2007] Pérez y Pérez, R. 2007. Employing emotions to drive plot generation in a computer-based storyteller. *Cognitive Systems Research* 8(2):89–109.

[Pérez y Pérez 2015] Pérez y Pérez, R. 2015. A computer-based model for collaborative narrative generation. *Cognitive Systems Research* 36:30–48.

[Scirea et al. 2015] Scirea, M.; Barros, G. A.; Shaker, N.; and Togelius, J. 2015. Smug: Scientific music generator. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 204.

[Sharples 1999] Sharples, M. 1999. How we write. *Writing as creative design*. London and New York: Routledge.

[Tobing and Manurung 2015] Tobing, B. C. L., and Manurung, R. 2015. A chart generation system for topical metrical poetry. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 308.

[Toivanen et al. 2013] Toivanen, J. M.; Järvisalo, M.; Toivonen, H.; et al. 2013. Harnessing constraint programming for poetry composition. In *Proceedings of the International Conference on Computational Creativity*, 160–167.

[Watanabe et al. 2017] Watanabe, K.; Matsubayashi, Y.; Inui, K.; Nakano, T.; Fukayama, S.; and Goto, M. 2017. Lyrisys: An interactive support system for writing lyrics based on topic transition. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 559–563. ACM.

Unified Classification and Generation Networks for Co-Creative Systems

Kunwar Yashraj Singh, Nicholas Davis, Chih-Pin Hsiao, Ricardo Macias, Brenda Lin, Brian Magerko

College of Computing
Georgia Institute of Technology
Atlanta, GA, USA

{kysingh, ndavis35, chsiao9, rmacias3, blin15, magerko}@gatech.edu

Abstract

This paper reports on a new deep machine learning architecture to classify and generate input for co-creative systems. Our approach combines the generational strengths of Variational Autoencoders with the image sharpness typically associated with Generative Adversarial Networks, thereby enabling a generative deep learning architecture for training co-creative agents called the Auxiliary Classifier Variational Autoencoder (AC-VAE). We report the experimental results of our network’s classification accuracy and generational loss on the MNIST numerical image dataset and TU-Berlin sketch data set. Results indicate our technique is effective for classifying and generating sketched object images, with larger sizes. We also describe how our network is particularly useful for co-creative agents since it can generate diverse concepts, as well as transform and morph user generated sketches while maintaining their concept identity.

Introduction

Three distinct trends are emerging in the field of computational creativity that form a spectrum of computational intervention in the creative process of users. On one end of the spectrum, intelligent systems assist users to achieve their creative goals with creativity support tools (CSTs). At the other end of the spectrum, users control different algorithmic parameters and deploy fully autonomous systems that generate artworks (i.e. procedural and algorithmic art generation). Computer colleagues represent the midpoint on this spectrum since these creative systems introduce co-creative agents in the user’s creative process as a collaborator or partner that can modify and add to a shared creative product.

Collaboration is a powerful way to inspire and support creativity. During creative improvisational collaborations, a new form of distributed creativity arises that can lead to emergent, dynamic, and unexpected meaning to support creativity in new ways (Sawyer and DeZutter 2009). We have seen evidence of how collaboration leads to dynamic and emergent meaning structures that inspire novel ideas during empirical studies of pretend play (Davis, Comerford, et al. 2015) and collaborative drawing (Davis, Hsiao, Yashraj Singh, et al. 2016). We have also found how collaborators often provide unexpected ideas and thus lead to surprising



Figure 1. Generated sketches from the TU-berlin sketch data test set using the AC-VAE network.

results. The same dynamism and flexibility that make creative collaboration so effective also make it challenging to implement autonomous co-creative agents.

We designed the Drawing Apprentice as a co-creative drawing partner to help explore what technical approaches and interaction designs are effective for facilitating creative collaboration in drawing. The system analyzes the user’s input and responds with its own contribution on a shared canvas. Our findings highlight the importance of classification and generative models for such systems, in order to recognize what type of object the user is drawing and generate an object in response. This creative dialogue of progressively adding related objects to a canvas can help the user generate more novel ideas and stay motivated to continue adding to the drawing. However, training a generative model with highly variable data is problematic in our case. The open-ended nature of drawing means users can introduce virtually any idea, and they expect real time responses. Furthermore, users are more interested in engaging in the drawing activity than explicitly training a system. These factors impede interactive machine learning and place a higher burden on the

interface and UX design to convince users to provide enough feedback to meaningfully train the system.

Creative domains, such as drawing, have two big knowledge engineering challenges: 1) The domain is *open-ended*, *dynamic*, and *highly improvisational*, which means the system needs to both classify and generate sketches in real time; and 2) creative domains do not have a lot of publicly available datasets. Furthermore, only a subset of the publicly available creativity data include data about the process of their creation. For example, in the domain of drawing, the TU-berlin sketch dataset is one of the only publicly available datasets of human sketched objects that contains stroke-level information.

The solution we propose is a deep learning architecture that can learn the latent distribution of example images to effectively classify and generate diverse instances of the learned concept. Our approach is called the Auxiliary Classifier Variational Autoencoder (AC-VAE). Our approach has two main benefits: 1) it enables the design of deeper Variational Autoencoders, and 2) it allows a training methodology to ensure that these networks do not collapse when trained on data that has high variance, such as sketched objects. The network employs a greedy training process for optimization, which ensures that these deeper networks do not collapse when training on data that has higher amount of variations.

We demonstrate how this network can effectively use variational autoencoding on large highly variable sketched image inputs and represent variations in the data by matching it to a unit Gaussian. This means if the latent vector is varied by sampling from a normal distribution, the examples smoothly morph to form each other. In the case of Drawing Apprentice, the network can take a sketch input, convert it to the latent variable, and then transform it to something else (within the same concept) by sampling on the latent vector. Our method uses one deep unified network to achieve high classification accuracy and low generation loss of image based data. This paper describes the AC-VAE network and the details of experimental results showing its efficacy on standard ML datasets and sketched object data. We also describe how the algorithm fits into the Drawing Apprentice architecture and how users will interact with it in the real-time application.

Related Work

Recent advances in deep machine learning enabled powerful classification and generation capabilities. Machine learning has been applied to traditional CSTs by improving the classification of the user's actions to provide better contextual support (Hsiao 2015). Similarly, deep learning has been applied to generative systems to produce extremely detailed and aesthetically pleasing artistic products, as demonstrated by the proliferation of neural style blending applications (Gatys, Ecker, and Bethge 2016). However, designing deep learning architectures for co-creative agents presents unique and interesting challenges as mentioned. These systems

need to be able to learn from diverse examples on the fly during improvisation to help facilitate a more seamless collaboration experience.

There have been some creative approaches for generating datasets in creative domains, such as crowdsourcing and pulling creative content from the web (Chen et al. 2014; Colton, Goodwin, and Veale 2012; Veale 2012). While these methods can be successful in generative computational creativity systems, co-creative systems have to actively improvise with users that can generate responses in real time with which the system is completely unfamiliar. This type of improvisational learning requires generalizing from a set of training examples that may have a high variability within the examples.

Recent advances in generative models have enabled algorithms to learn a distribution from input examples and generate new examples from them, which can help train deep neural networks. Some popular methods for generating examples include Autoencoders (Vincent et al. 2010; Bengio et al. 2013), Generative Adversarial Networks (GAN) (Radford, Metz, and Chintala 2015), and Variational Autoencoders (VAE) (Kaae Sønderby et al. 2016; Johnson et al. 2016; Walker et al. 2016). Autoencoders are unsupervised networks that consists of an encoder and decoder stitched together that learn to reconstruct the inputs provided to the network. Generative Adversarial Networks learn the input distribution by training in an adversarial fashion. GANs consists of a discriminator and generator module that attempt to constantly fool each during the training process in order to learn the distribution (Goodfellow et al. 2014). While GANs have gained in popularity recently due to their effectiveness at learning the input distribution to generate realistic images, their stability remains an open question as training them effectively requires a lot of tuning (Arjovsky and Bottou 2017; Salimans et al. 2016). These networks are difficult to train on large image sizes, and often do not converge on large and complex image inputs (Goodfellow et al. 2014).

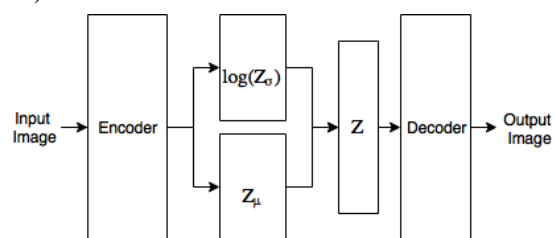


Figure 2. A typical Variational Autoencoder

Variational Autoencoders are a variant of autoencoders that learn a compact representation of the input space, referred to as the latent space (Sønderby et al. 2016). The latent space learned by VAEs are rich in a sense that they also encode various properties of the image implicitly, which is useful for performing vector arithmetic to generate new images (Sønderby et al. 2016). The downside of this approach

is that the images are blurry when compared to GAN due to the mean loss being optimized. Additionally, it is not feasible to train a VAE that is deep and works with large image sizes (Arjovsky and Bottou 2017; Salimans et al. 2016). Recent techniques such as Ladder Variational Autoencoders (LVAEs), Importance Weighted Autoencoders, and Matryoshka Networks have tried to address this problem by utilizing warm up training to introduce the variational term gradually (Bachman 2016; Sønderby et al. 2016; Burda, Grosse, and Salakhutdinov 2015). There has also been success using hybrid top down and bottom up networks (Bachman 2016). In this paper, we build on the finding of LVAE and Matryoshka architecture to overcome some of the limitations of VAEs. This approach enables training deeper and larger VAEs and opens the possibility of handling large input images and leading to sharper output results. We will describe the overall system components before discussing the details of our approach and how it can fit into the system.

System Description

Drawing Apprentice System

The Drawing Apprentice is implemented as a web application (He et al. 2016; Davis, Hsiao, Yashraj Singh, et al. 2016; Davis, Hsiao, et al. 2015) with a client-server architecture that enables multiple people to collaborate on the same drawings as well as the co-creative agent. It was designed for use with stylus- or touch-based interactions, but a mouse can also be used. To briefly summarize its function, the system takes user input lines, transforms those lines based on the sketch recognition and generation algorithms, and outputs new lines onto the same canvas. Unique and defining features of input line sets are determined by clustering the data points and sending that them into the neural network. This allows the neural network to derive its own classifications scheme based on the data it has been given.

As shown in Figure 3, the system was seeded with various algorithms: (1) line transformation functions, such as trans-

lation, scaling, rotation (Davis et al. 2011); (2) line morphing techniques that change the individual features of the input lines to create new lines that retain a similarity to the input lines (Davis, Hsiao, et al. 2015; Davis et al. 2014); and (3) recognizing the sketching object and generating similar object in response (Davis, Hsiao, Singh, et al. 2016). During drawing collaboration, the user may begin at any point, which can lead to synchronous collaboration. When the user pauses the drawing, the agent will recognize it as a “turn”, and adopt one of the algorithms to generate the response lines. When the user is not satisfied with the agent’s drawing actions, she could provide feedback on them to optimize the system’s model by clicking on the up and down voting buttons. To simulate the dynamism and embodied nature of real-time human collaboration, the Drawing Apprentice character draws lines dynamically, meaning lines do not appear at once in full, but are gradually animated through until their completion. Dynamic line drawing is meant to provide a sense that the system is going through the embodied act of creating a line. This presents an interesting opportunity where improving the user experience design might potentially improve the performance of the machine learning algorithms (since more feedback helps train the system).

One of the primary limitations of the Drawing Apprentice system that requires the proposed architecture is the ability to generate diverse instances of learned objects. Before integrating the new architecture, the system’s line generation capabilities were restricted to reacting to the user’s line or recognizing groups of lines as objects and selecting a related object from its databased of known concepts and drawing the selection directly. The system never actively generated new instances of concepts that it learned. The new AC-VAE network learns the latent distribution of example inputs, which allows sampling in that space to generate new outputs based on that distribution. We define two types of sampling for our use case.

Zero-Sampling: when the latent vector of the input image is sampled close to the mean having 0 standard deviation.

Tail-Sampling: when the latent vector of the input image is sampled way from the mean towards the tail of the distribution, having standard deviation close to 1.

Based on the two sampling types we can indicate the amount of variation required on the input. In case of zero-sampling, the input image is reconstructed as close to itself as possible whereas in tail-sampling, the input is varied significantly while still retaining some of the input features.

The following sections describe the design decision and experimentations related to the new network architecture.

Network Architecture

When initially tackling the sketch generation problem using Bayesian Program Learning (BPL)(Lake, Salakhutdinov, and Tenenbaum 2015), we found that the quantification of the semantic representation of differing sketch types is an intractable problem due to the variations in size, detail, and

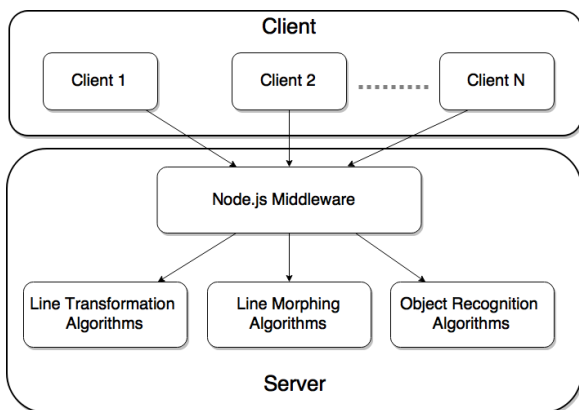


Figure 3. The Drawing Apprentice system overview.

primitives inherent in each sketch. The large number of combinations of stroke orderings, along with their semantic significance for each sketch archetype is very difficult to represent in a way BPL can generate examples. The mathematical representation of sketch archetypes would be the ideal basis to use when generating new sketch examples. But due to the intractability of the problem, we decided that a more suitable framing of the problem would be an approximation of a family of posterior distributions. These distributions should be similar to a Gaussian, so that variation between sketches of the same type can be captured. The goal is to then sample from these distributions, which will also be representative of the vector space from which sketches can be generated, with a given standard deviation to produce a variation on a given image. We wish to represent distributions that are similar enough to the true generative distribution so that generated images are semantically coherent, but not so closely related that a classifier being trained on a generated example would receive little increase in classification accuracy. Therefore, it is necessary to use a model that can approximate these distributions based on the hidden representation of the salient variables present in each sketch, and this narrowed down the family of algorithms that would be appropriate for the task quite a bit.

We noticed that we could build a modified Variational Autoencoder that would be suitable for the task of approximating the family of posterior distributions because we could build an architecture which uses hierarchies of conditional random variables to represent them (Sønderby et al. 2016; Bachman 2016). The salient aspects of each sketch, represented in the latent space, are used to condition the distributions.

A typical VAE architecture consists of an encoder network and a decoder network. The encoder network takes in the input image and maps it to a latent vector Z of predefined length. This latent vector is then used by the decoder to reconstruct the input image given the information contained in it (Doersch 2016). The variational part comes into play when the encoder does not directly encode the image but splits it into two vectors of same length, namely Z_μ and $\log(Z_\sigma)$. Then, the final encoding, which is the latent vector Z , is formed by sampling on the learned mean and standard deviation (Doersch 2016; Nowozin, Cseke, and Tomioka 2016). For example: $Z = Z_\mu + e^{\log(Z_\sigma)} * \epsilon$ where $\epsilon \sim N(0,1)$ where N is the standard normal distribution. This separation of latent space into two components is referred to as the ‘reparameterization’ trick (D. P. Kingma and Welling 2013; Doersch 2016) that helps in applying KL-divergence in order to evaluate how well the latent variable matches the unit Gaussian (D. P. Kingma and Welling 2013; Nowozin, Cseke, and Tomioka 2016; Doersch 2016). Hence, the loss function that VAE tries to optimize as its objective is (1) The reconstruction loss and (2) the KL-divergence (D. P. Kingma and Welling 2013).

$$Loss_{reconstruction} = BinaryCrossEntropy(Actual, Generated)$$

$$Loss_{KL} = -\frac{1}{2} Mean(1 + \log(Z_\sigma) - Z_\mu^2 - e^{\log(Z_\sigma)})$$

$$Loss = Loss_{reconstruction} + Loss_{KL}$$

Previous work has shown that smaller networks are able to optimize the loss function easily but cap off after a point due to the difficulty smaller networks face when attempting to capture highly non-linear relationships in data. Deeper networks on the other hand tend to get stuck in a local maxima and collapse when trained on large image sizes (Sønderby et al. 2016; Bachman 2016; Burda, Grosse, and Salakhutdinov 2015). The LVAE architecture demonstrated that training deeper Multilayer Perceptron layers requires gradually introducing the KL term in the loss function during the training process. However, this method had limitations, particularly in knowing at what epoch the KL term should be introduced and on what scale (Sønderby et al. 2016). In the literature, these decisions are usually determined by experimentation and fine-tuning.

The network discussed in this paper was built to generate as well as classify images using the learned features. We call this architecture Auxiliary Classifier Variational Autoencoder (AC-VAE) that consists of an auxiliary classifier as part of the encoder along with the latent vector. We build upon findings from LVAE and use deep residual learning to construct a very deep model that can work with large image sizes and train efficiently in order to generate and classify images. Classification networks built using residual blocks are known for being very deep (>30 layers). Their ability to adjust themselves depend on the degree of linearity of the data (He et al. 2016) eases the training process. These residual blocks, as shown in Figure 4, form an integral part of our network and are symmetrical for the encoder and decoder network. They make use of Batch Normalization, which speeds up training and can be incorporated into deeper networks without running the risk of overfitting. The diagram shown in Figure 5 shows the overall network architecture of AC-VAE using residual blocks and identity mappings between the convolutional layers for the sketch dataset.

Training deeper convolutional networks to optimize the VAE loss requires significant tuning and usually the KL-term becomes very large in the first few epochs to account for variations in the data (Sønderby et al. 2016; Doersch 2016). Hence, we came up with a training strategy that could potentially overcome these limitations and train the deep network effectively. To do so, we reexamined the objective that the network was trying to optimize and found something similar to findings mentioned by LVAEs but the strategy had to be changed. We found that if we minimized just the reconstruction error before we moved to minimize the overall loss, the network would be able to learn good features prior to minimizing the variational error. This scheme enabled the network to partially encode the data distribution before moving on to learning the variations. Apart from this,

we also introduced an auxiliary classifier that shared the

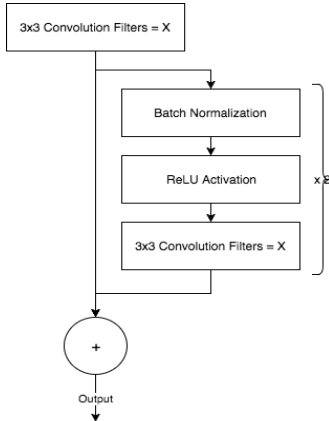


Figure 4. Residual block with identity mapping, used as the basis of our network where X denotes the number of filters used in each convolutional layer.

learned features to classify the input data. Therefore, the overall loss term that AC-VAE minimizes is:

$$Loss_{classification} = \text{CategoricalCrossEntropy}(\text{Labels}_{true}, \text{Labels}_{predicted})$$

$$Loss_{AC-VAE} = Loss_{reconstruction} + Loss_{KL} + Loss_{classification}$$

Furthermore, we noticed that the $Loss_{KL}$ term acts as regularizer for our network, as mentioned in (Sønderby et al. 2016; Bachman 2016; D. P. Kingma and Welling 2013), and helps the entire network counter overfitting. The process below outlines the way the network is trained.

Method: Train AC-VAE

For each epoch:

1. Train the network once to minimize $Loss_{reconstruction} + Loss_{classification}$
2. Train the network for k epochs to minimize $Loss_{AC-VAE}$

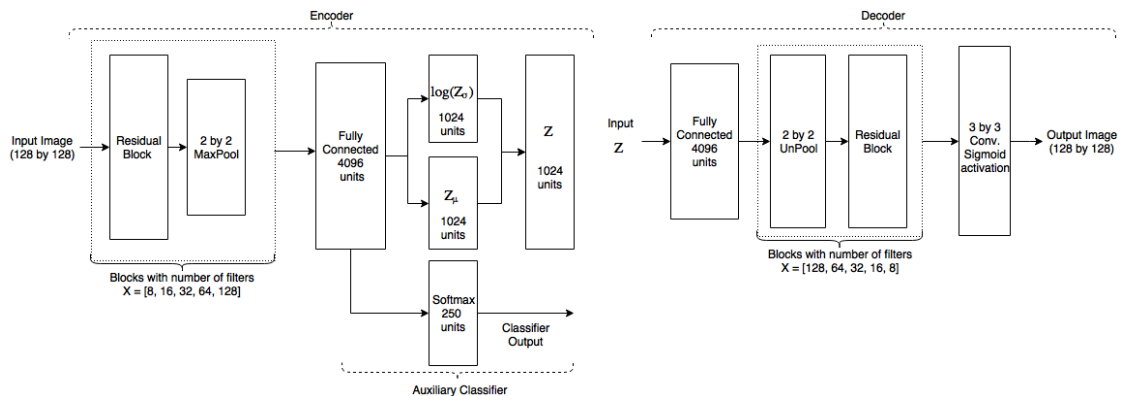


Figure 5. AC-VAE used for sketch data. The auxiliary classifier uses the features extracted by the encoder to classify.

This method of training facilitates the network to learn good weights before trying to learn the variation across the image batch and this is particularly helpful for deeper network that have large number of parameters. The $Loss_{KL}$ term acts as a regularization term (D. P. Kingma and Welling 2013; Doersch 2016) and counters the network from overfitting to the training data. A more intuitive way to reason about the training process is that the first sub-epoch tries to match the training distribution by training the network as a deterministic auto-encoder whereas, the second sub-epoch tries to pull the network away from the distribution by introducing variational cost. This enables the network account for variations in the training data and represent it within the latent vector. This way of greedily minimizing the pieces of the overall loss within each iteration stops the network from collapsing when there are large deviations in the training data and when the minimization process suddenly introduces large variances in the KL-term.

Experiments and Evaluation

MNIST

MNIST dataset contains a collection of handwritten digit images from 0 to 9 (image size 28 by 28 pixels) with 60,000 images in the training set and 10,000 in the test set. (LeCun, Cortes, and Burges 2010). This dataset is widely used to benchmark generative as well as classification networks. For our experiments with MNIST, we created a network of 2 residual blocks as the encoder and 2 for the decoder to account for the non-linearity.

The auxiliary classifier was attached to the intermediate dense layer right before the latent layer and it consisted of another residual block as shown in Figure 4. The network mentioned in the paper was trained with Adam optimizer (D. Kingma and Ba 2014), using no data augmentation and with a learning rate of 0.001 as used during training LVAE (Sønderby et al. 2016). We ran the experiments for different dimensions of latent vectors from 2, 50 and 100 for 50 epochs.

The training process use the training methodology as outlined in this paper and we set $k = 2$. During the training process, we monitored the log-likelihood to report the generation accuracy in addition to the classification accuracy measured. Table 1 below reports the best generation loss achieved in addition to the auxiliary classification accuracy on the MNIST test set and compares it to the other state-of-the-art methods.

We can see that AC-VAE can minimize the loss better than the previously employed methods, this is because network can adjust itself to be as linear or nonlinear as possible depending on the training data. To add to it, the high number of convolutional feature extraction layers work well with spatial data, have more parameters and can better represent the distribution. Apart from the generative mode, we can see that the features learned by the intermediate layer of the network are even useful for classification purposes and the auxiliary classifier reaches a good accuracy of 99.31 percent, which is comparable to the state-of-the-art classification results on MNIST dataset without any data augmentation.

Model	Generation Log Likelihood	Classification Accuracy
VAE, 2-layer + VGP (Sønderby et al. 2016; Tran, Ranganath, and Blei 2015)	-81.90	-N/A
LVAE, 5-Layer + fine-tuning (Sønderby et al. 2016)	-81.84	-N/A
LVAE, 5-Layer + fine-tuning + IW=10 (Sønderby et al. 2016)	-81.74	-N/A
MATNET (Bachman 2016)	-80.5	-N/A
AC-VAE, Latent size = 50	-55.31	99.22%
AC-VAE, Latent size = 100	-52.63	99.31%

Table 1. Results on the MNIST test set. Generation accuracy is measured using log-likelihood as used in previous works.

TU-Berlin Sketch Dataset

The TU-Berlin dataset is a collection of human drawn sketches of objects from everyday life. It is one of the first datasets to contain several exemplar sketches for a wide variety of human-drawn concepts. The dataset contains 250 categories, with each category containing 80 distinct sketches for a total of 20,000 images. This dataset was selected because it aligns with the scope of Drawing Apprentice project, and it is one of the largest collection of human drawn sketches. We were motivated to test the network and see how well the network could understand and generate

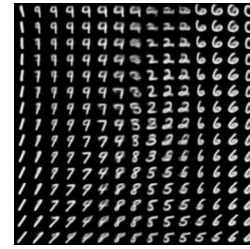


Figure 6. 2D Latent space representation of MNIST test images. Images are generated by sampling points from $[0, 0]$ (top left) to $[1, 1]$ (bottom right). Each row goes from zero-sampling to tail sampling on the right.

variations of the sketches present in this dataset. If successful, this would help the program generate unique variations to the sketches drawn by the user depending on how far the

Model	Generation Log Likelihood	Classification Accuracy
DRAW + VGP (Tran, Ranganath, and Blei 2015) (300 epochs)	-423.9	-
AC-VAE (100 epochs)	-887	39%

Table 2. Results on the TU-Berlin test set where generation accuracy is measured using log-likelihood.

algorithm samples from the distribution, adding transformational qualities to the agent.

For our purpose, the sketch images were resized to 128 by 128 pixels, and we tweaked the network to handle larger images by using 5 residual blocks for the encoder and 5 for the decoder. The dense layers that bridged the encoder and decoder consisted of 4096 neurons in the intermediate layer and 1024 in the latent layer. Overall, the network consisted of 45 layers and was trained using ADAM optimizer with a learning rate of 0.00001 for 100 epochs.

To test our model, we split the 20,000 images into training and test set with 18,000 images in the training set and the remaining 2,000 in the test set. This split was chosen because previous evaluation methods used the same split to benchmark the models. The above table reports the generation and classification accuracy we achieved using our model and compare it to DRAW + VGP as used in D. Tran et al. (Tran, Ranganath, and Blei 2015). Though, DRAW + VGP generated using window sizes, the method cannot be directly compared to ours as it uses a sliding window approach instead of generating end to end. Table 2 above shows the accuracy for 300 epochs for DRAW + VGP and for 100 epochs for AC-VAE. Few of the generated images are also presented in figure 1 and 7 where figure 7 highlights the variational part of the generative network where the first image is the input image used to obtain the initial latent vector and is sampled upon to get the generated images. The

sampling is closer to mean on the left and away from the mean as we move right.

Discussion

Our experimental results show that AC-VAE is effective for classifying and generating sketched object images as required by the Drawing Apprentice co-creative system. This approach combines the strength of VAEs with the image sharpness typically associated with GANs. The network leverages the strength of VAEs to represent variations in the data by matching it to a unit Gaussian. This means if the latent vector is varied by sampling from a normal distribution, the examples smoothly morph to form each other as seen in figure 6. This continuous representation of the conceptual space enables the system to generate diverse examples of that concept during the co-creation with the user, such as helping designers explore the conceptual space of their design.

The AC-VAE generation capabilities are useful for co-creative agents because it can generate concepts representing different degrees of variations within the overall concept. For example, the network can be used to intelligently alter user’s sketched objects by converting it to the latent variable and then transforming it to something else within the same concept by sampling on the latent vector.

Future work can also explore combining the latent vectors of two objects into a new space to enable object blending. For example, combining the concepts of tree and airplane may produce interesting tree airplanes when sampling the latent vector. This type of object blending can be used in the real time co-creative system by turning one object into another resulting in a conceptual shift for the user. These types of conceptual shifts are a unique part of the creative process, and are particularly relevant to collaboration as different perspectives often reveal new ways of seeing problems and opportunities in the environment.

Conclusions

This paper reported on a new deep machine learning architecture to classify and generate input for co-creative systems. This network combines the generational strengths of Variational Autoencoders with the image sharpness typically associated with Generative Adversarial Networks, thereby enabling a generative deep learning architecture for training co-creative agents called the Auxiliary Classifier Variational Autoencoder (AC-VAE).

Our approach has two benefits: 1) it enables the design of deeper Variational Autoencoders, and 2) it allows a training methodology to ensure that these networks do not collapse when trained on data that has high variance, such as sketched objects. We reported the experimental results of our network’s classification accuracy and generational loss on the MNIST numerical image dataset and TU-Berlin sketch data set. Results indicate our technique is effective for classifying and generating sketched object images, even

with larger image size. We also described how our network is particularly useful for co-creative agents since it can generate diverse concepts, as well as transform and morph user generated sketches while maintaining their concept identity.

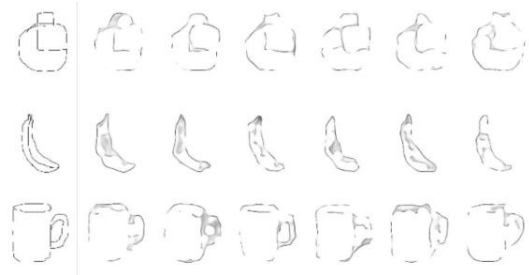


Figure 7. Variations of examples from Tu-Berlin test data generated using AC-VAE. Image on the far left represents the sketch image that was fed into the system. The variations are generated by randomly sampling on the latent variable of size 1024 used in the experiment.

Acknowledgements

This work was supported by NSF Grant IIS-1320520.

References

- Arjovsky, Martin, and Léon Bottou. 2017. “Towards Principled Methods for Training Generative Adversarial Networks.” In *NIPS 2016 Workshop on Adversarial Training. In Review for ICLR*. Vol. 2016.
- Bachman, Philip. 2016. “An Architecture for Deep, Hierarchical Generative Models.” In *Advances in Neural Information Processing Systems*, 4826–4834.
- Bengio, Yoshua, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. “Generalized Denoising Auto-Encoders as Generative Models.” In *Advances in Neural Information Processing Systems*, 899–907.
- Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov. 2015. “Importance Weighted Autoencoders.” *arXiv Preprint arXiv:1509.00519*.
- Chen, Quanze, Chenyang Lei, Wei Xu, Ellie Pavlick, and Chris Callison-Burch. 2014. “Poetry of the Crowd: A Human Computation Algorithm to Convert Prose into Rhyming Verse.” In *Second AAAI Conference on Human Computation and Crowdsourcing*.
- Colton, Simon, Jacob Goodwin, and Tony Veale. 2012. “Full Face Poetry Generation.” In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.
- Davis, Nicholas, Margeaux Comerford, Mikhail Jacob, Chih-Pin Hsiao, and Brian Magerko. 2015. “An Enactive Characterization of Pretend Play.” In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 275–284. ACM.

- Davis, Nicholas, Ellen Yi-Luen Do, Pramod Gupta, and Shruti Gupta. 2011. "Computing Harmony with PerLogicArt: Perceptual Logic Inspired Collaborative Art." In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, 185–194. ACM.
- Davis, Nicholas, Chih-Pin Hsiao, Kunwar Yashraj Singh, Lisa Li, Sanat Moningi, and Brian Magerko. 2015. "Drawing Apprentice: An Enactive Co-Creative Agent for Artistic Collaboration." In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 185–186. ACM.
- Davis, Nicholas, Chih-Pin Hsiao, Kunwar Yashraj Singh, and Brian Magerko. 2016. "Co-Creative Drawing Agent with Object Recognition." In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.
- Davis, Nicholas, Chih-Pin Hsiao, Kunwar Yashraj Singh, Lisa Li, and Brian Magerko. 2016. "Empirically Studying Participatory Sense-Making in Abstract Drawing with a Co-Creative Cognitive Agent." In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, 196–207. ACM.
- Davis, Nicholas, Yanna Popova, Ivan Sysoeven, Dingtian Zhang, and Brian Magerko. 2014. "Building Artistic Computer Colleagues with an Enactive Model of Creativity." In *International Conference on Computational Creativity, Ljubljana Slovenia*. AAI.
- Doersch, Carl. 2016. "Tutorial on Variational Autoencoders." *arXiv Preprint arXiv:1606.05908*.
- Gatys, Leon A, Alexander S Ecker, and Matthias Bethge. 2016. "Image Style Transfer Using Convolutional Neural Networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423.
- Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Nets." In *Advances in Neural Information Processing Systems*, 2672–2680.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.
- Hsiao, Chih-Pin. 2015. "SolidSketch: Toward Enactive Interactions for Semantic Model Creation." In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 329–330. ACM.
- Johnson, Matthew J, David Duvenaud, Alexander B Wiltchko, Sandeep R Datta, and Ryan P Adams. 2016. "Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders." *arXiv Preprint arXiv:1603.06277*.
- Kaae Sønderby, Casper, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. "How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks." *arXiv Preprint arXiv:1602.02282*.
- Kingma, Diederik, and Jimmy Ba. 2014. "Adam: A Method for Stochastic Optimization." *arXiv Preprint arXiv:1412.6980*.
- Kingma, Diederik P, and Max Welling. 2013. "Auto-Encoding Variational Bayes." *arXiv Preprint arXiv:1312.6114*.
- Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. "Human-Level Concept Learning through Probabilistic Program Induction." *Science* 350 (6266): 1332–1338.
- LeCun, Yann, Corinna Cortes, and Christopher J.C. Burges. 2010. "The MNIST Database of Handwritten Digits." <http://yann.lecun.com/exdb/mnist/>.
- Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka. 2016. "F-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization." In *Advances in Neural Information Processing Systems*, 271–279.
- Radford, Alec, Luke Metz, and Soumith Chintala. 2015. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *arXiv Preprint arXiv:1511.06434*.
- Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. "Improved Techniques for Training Gans." In *Advances in Neural Information Processing Systems*, 2226–2234.
- Sawyer, R Keith, and Stacy DeZutter. 2009. "Distributed Creativity: How Collective Creations Emerge from Collaboration." *Psychology of Aesthetics, Creativity, and the Arts* 3 (2): 81.
- Sønderby, Casper Kaae, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. "Ladder Variational Autoencoders." In *Advances in Neural Information Processing Systems*, 3738–3746.
- Tran, Dustin, Rajesh Ranganath, and David M. Blei. "The variational Gaussian process." *arXiv preprint arXiv:1511.06499* (2015).
- Veale, Tony. 2012. "From Conceptual Mash-Ups to Bad-Ass Blends: A Robust Computational Model of Conceptual Blending." In *Proceedings of the Third International Conference on Computational Creativity*, 1–8.
- Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion." *Journal of Machine Learning Research* 11 (Dec): 3371–3408.
- Walker, Jacob, Carl Doersch, Abhinav Gupta, and Martial Hebert. 2016. "An Uncertain Future: Forecasting from Static Images Using Variational Autoencoders." In *European Conference on Computer Vision*, 835–851. Springer.

Déjà Vu All Over Again

On the Creative Value of Familiar Elements in the Telling of Original Tales

Tony Veale

School of Computer Science and Informatics
University College Dublin, Belfield D4, Ireland.
Tony.Veale@UCD.ie

Abstract

Movie studios have compelling reasons to love sequels. Familiar characters from successful films are valuable *properties* that come with large built-in audiences eager to pay for more. That such characters are commodities is beyond dispute, yet they are as much commodities for creative story-telling as for commercial film-making. Familiar characters come with pre-existing audiences and pre-existing audience *expectations*, and writers can exploit the latter to reduce exposition, establish *mise en scène*, create mood or motivate the use of genre tropes. Familiarity can also be abused for comic ends, to create narratives dense in references to other stories, worlds or genres. Post-modern irony thus abounds in stories that combine old characters in new, clever and perhaps even logically impossible ways. In this work we explore the value of a large knowledge-base of familiar characters within the plotting mechanics of the *Scéallextric* system, to quantify the extent to which familiarity can enhance or diminish our enjoyment of machine-crafted stories.

Tarzan vs. IBM

Jean-Luc Godard's first choice for the name of his seminal 1965 Sci-Fi film *Alphaville* was not the name of the film's dystopian cityscape, but *Tarzan Versus IBM*. While those characters are familiar in themselves – clichéd, even – the combination is unexpected, jarring and resonant. Godard's film is ultimately a bravura exercise in semiotics, and he uses his characters as much for their potential to signify as their potential to anchor a plot. So *Tarzan* signifies all that is natural, intuitive, vigorous and rooted in the world, while *IBM* signifies largely the opposite, that is, the mechanical, the logical and the tightly controlled. For Godard these two characters represent the antagonistic forces shaping France in the mid-60s, and so his film sets out to capture the battle between scientific technocracy and romantic freedom. But as his chosen signifiers raised obvious legal issues, Godard eventually chose two other familiar characters to carry his narrative. His *Tarzan*-substitute would be *Lemy Caution*, a grizzled detective in a popular series of hard-boiled novels, and his ersatz *IBM* would be a mix of *Wernher Von Braun*, *John Von Neumann* and *Robert Oppenheimer*, whom he blended into his nefarious scientist 'Nosferatu' *Von Braun*.

Godard exploited a strategy that was cheekily articulated in Flann O'Brien's 1939 comic novel *At Swim Two Birds*:

“Characters should be interchangeable as between one book and another. The entire corpus of existing literature should be regarded as a limbo from which discerning authors could draw their characters as required, creating only when they failed to find a suitable existing puppet. The modern novel should be largely a work of reference. Most authors spend their time saying what has been said before – usually said much better. A wealth of references to existing works would acquaint the reader instantaneously with the nature of each character.”

William S. Burroughs (1963) famously used the corpus of existing literature as fodder for his *cut-up method*, wherein new texts were formed from old via the slicing, dicing and random re-splicing of text chunks. However, his approach lacks a certain finesse that does not allow creators to focus on niceties such as characterization or plot. What is needed for a story-generation process that respects these notions is a knowledge-base, to turn the *limbo* of existing characters into a well-structured inventory of foibles and affordances.

We use one such knowledge-base in this work, the *NOC Non-Official Characterization List* of Veale (2015a, 2016). The *NOC* provides copious detail on over 800 characters drawn from diverse genres and contexts that are not limited to the wholly fictional. Comprising 30,000 semantic triples the *NOC* affords story-tellers a level of lively detail for the construction of high-level scenarios and individual scenes that – we hypothesize – can transform a skeletal plot into a vivid narrative that is at once both novel and familiar. For Giora *et al.* (2004) argue that a complex stimulus – such as a headline, a joke or a story – is *optimally innovative* when it allows readers to contrast a creator's unconventional use of familiar elements, such as words and names, with their more conventional (or “salient”) uses. Familiar elements, like familiar faces, always provoke a salient response even if this response is subsequently reassessed as incongruous (Giora, 1997). We use the *NOC* here to ensure that stories generated by our system are optimal innovations that flit along the thin line dividing the educated from the insolent.

Like *Tarzan & IBM*, our pre-existing characters are chosen for this potential to signify larger themes and ideas,

such as the conflict between *rich & poor, strong & weak or logical & illogical*. Fictional characters are liberally mixed with real or historical figures in pairings such as *Steve Jobs & Leonardo da Vinci, Frank Underwood & Richard Nixon* or *Lex Luthor and Donald Trump*. But these characters can only come to life in stories with plots that put them to good use. These plots are crafted by a system named *Scéalextric*, which assembles skeletal plots from *action n-grams* (akin to the *Web n-grams* of Brants & Franz, 2006) that resemble the clickable segments of a toy slot-car racing track (Veale, 2016). ‘*Scéalextric*’ is a portmanteau of ‘*scéal*’, the Irish word for ‘story’, and *Scalextric*, a brand of toy racing kits. Our focus here is on conceptual integration – in the sense of Fauconnier & Turner (1998;2002) – to blend characters from the NOC with plots built using *Scéalextric*, to show how the former can elevate our appreciation of the latter. Importantly, we make the NOC and *Scéalextric* databases available to the community to support further research.

Mechanical story generation pre-dates the birth of the modern computer, so our story begins in the next section with a discussion of the relevant background to this work. We then present the NOC and its contents in greater detail, before exploring the *Scéalextric* model of plot generation. The NOC is then integrated with *Scéalextric* in a fashion that affords creativity from the highest to the lowest levels of a story, before the fruits of this integration are evaluated. The paper concludes with some indications of future work.

Related Work and Ideas

The 1920s was a fertile period for the structural analysis of narrative, producing analyses that would shape and reflect the future automation of the story creation process. In 1928 Vladimir Propp published his *Morphology of the Folktale*, offering an influential structuralist view into the familiar elements that dictate the structure of novel stories. Propp treated tales as akin to cocktails in a trendy bar; each may follow a different recipe and impart a distinctive taste, yet each is drawn from the same set of familiar ingredients. A folklorist and empiricist, Propp built his morphemic system of recurring story *functions* from a painstaking analysis of a corpus of Russian tales. His work influenced generations of folklorist analysis to come, but it has also found favour with computationalists who seek to model story-telling as an act of combinatorial creativity (see Gervás 2013; Veale, 2014). The character functions of Propp’s morphology are not specific characters *per se*, but familiar archetypes of which memorable characters such as *Tarzan, Batman, Neo, Bond, Bourne, Loki, Vader* and so on are vivid instances.

While Propp’s system is focused on analysis, 1928 also saw the publication of a more practical, generation-focused structuralist approach in William Wallace Cook’s *PLOTTO*. Cook was driven not by academic curiosity but by the need to produce novel tales of his own to a punishing schedule, sometimes writing a new book in a week. To systematize the generation of plots for each new tale, Cook compiled a corpus of ~1500 *master plots*, each comprising three parts or *clauses*. These plots are numbered and comprehensively categorized and cross-indexed to allow *PLOTTOist* writers

to quickly flesh a skeletal plot into a fuller outline. Though Cook’s tripartite plot skeletons are steeped in melodrama, *Plotto* anticipates much of 60s/70s symbolic AI, especially that of the Schank & Abelson school (1977). Moreover, Cook’s use of tripartite plot skeletons that can be simply clicked together still has much practical relevance today.

Yet it was not Cook’s *Plotto* that would later find favour with Hollywood studios but Joseph Campbell’s Propp-like comparative analysis of the heroic tales of world religions, with his 1949 book *The Hero With a Thousand Faces*. In those tales Campbell saw evidence of a protean *mono-myth* that imposes a single deep-structure on a great many heroic myths: “*A hero ventures forth from the world of common day into a region of supernatural wonder: fabulous forces are encountered and a decisive victory is won: the hero comes back from this mysterious adventure with the power to bestow boons on his fellow man.*” Campbell’s work has been acknowledged as a major influence in the creation of the *Star Wars* mythos by creator George Lucas, the success of which persuaded film studios to pursue the structuralist ideas of Propp and Campbell. So when Vogler (1984/1998) disseminated a far-reaching memo on Campbell’s ideas, the schematic agenda became a commercial imperative.

Most AI systems that produce tales – as described in e.g. Meehan (1981), Turner (1994), Pérez y Pérez & Sharples (2004), Riedl & Young (2010) and Gervás (2013) – rely on abstract schematic structures in the mould of Propp, Cook, Campbell and Vogler: they model story-telling as a process of instantiating a core set of simple, reusable forms in new and diverse ways. Gervás *et al.* (2016) employed Proppian schemas to generate plot elements for musical theatre, the products of which were employed in a commercial musical. Veale (2014, 2016) gave Campbell’s notion of the hero’s journey a computational form in a generator of story *arcs* that models the transformation of a character over time. An arc specifies the start and end point of a character’s journey but does not fully articulate a path between those points. In this paper we aim to meld a character with its arc in a beat-by-beat sequence of specific actions that lays out this path. While this sequence resembles a random walk more than a premeditated plan in the sense of Riedl & Young, the walk is sufficiently constrained to yield a coherent storyline.

Schematic structures offer typed slots that may only be filled by characters of the matching types. Cook’s master plots use *A* and *B* to label slots for male and female fillers respectively, while Propp’s and Campbell’s schemas allow for a richer set of types to fill their slots. But the challenge – and the opportunity – for a story generator is to do more than fill slots with matching fillers. A good story requires a tight conceptual blend of characterization and plot, so that what the audience already knows about a given character can spill out of its local slot to colour the action as a whole. This is a challenge we address in the following sections, to allow the known affordances of a familiar character (what he/she wears, likes, dislikes, does for a living, etc.) to fully inform the instantiation and rendering of a plot action and thereby create a more memorable experience for the reader. We begin by taking inventory of these diverse affordances.

Persons of Interest

Godard reached into a grab-bag of cultural icons to pull out *Tarzan & IBM* as the ideal signifiers for his message. But when these proved legally cumbersome to use, he dipped in for another rummage, to now pull out *Lemy Caution* and *Wernher Von Braun* (not to mention *Robert Oppenheimer*, *John von Neumann* and *Nosferatu*). In other words, Godard used the body of shared cultural landmarks as a resource to drive his process of combinatorial creativity, in a cycle of selection, juxtaposition, rejection and repeated re-selection. He had his own artistic constraints to satisfy in each cycle: note, for instance, how the pairings of *Tarzan & IBM* and *Caution & Von Braun* each juxtapose a fictional creation with a real historical entity. Like the film as a whole, each pairing has one foot in reality and the other in pure fantasy. Different creators impose different constraints, and so our resource, a database of cultural icons, must be detailed and comprehensive if it is to satisfy many (if not all) of them.

Our NOC list will serve as this comprehensive resource. But what makes someone iconic enough to be worthy of representation in the NOC, and which aspects of this entity should the NOC aim to capture? The qualities that elevate a person into a cultural reference point are precisely those that seem to exist in a concentrated and exemplary form in that one person, yet which are so common in our shared experience as to be predicated of many others besides. Our ambitions for the NOC go beyond the representation of the simple adjectival qualities of an iconic person, and include their many affordances as complex, fully-realized entities in their own world, such as gender (*male* or *female*), locale (e.g. *New York*, *Tatooine*), style of dress, spouses or lovers, known enemies, apt vehicles, trademark weapons, relevant domains (e.g., *arts*, *science*, *politics*), semantic types (e.g., *politician*, *playboy*), fictive status (*fictional* or *real*), genres (e.g., *science fiction*), creators and screen actors (if fictive), typical activities (e.g., *building casinos*, *running political campaigns*), political leanings (*left*, *right* or *moderate*) and group affiliations (e.g. *Tony Stark* belongs to *The Avengers* and *Eliot Ness* belongs to *The Untouchables*; *Darth Vader* belongs to the *Dark Side* and *Donald Trump* to the *GOP*).

The affective content of the NOC is intended to provide ‘*talking points*’ in the sense of Veale & Hao (2008). That is, it codifies the qualities and behaviours that we humans naturally focus upon whenever we talk about a celebrity or compare one person to another. The NOC thus divides the adjectival qualities of each entity into positive and negative talking points. The former are those with a positive lexical affect, such as *resolute*, *wealthy* and *media-savvy*, the latter are those with an obvious negative affect, such as *evil*, *tight-fisted* and *devious*. The NOC provides at least three positive and three negative talking points for each of its 800+ entries, so that stories which are built around these entries can offer more than regurgitated clichés and instead offer nuanced qualities that motivate emergent inferences.

The NOC uses a standard frame format for its contents, allocating one frame per entity with its various slots and fillers as outlined above, and additional frames for those fillers that are themselves worthy of further elaboration.

Thus, for example, the NOC provides additional frames for one’s clothing, weapons and vehicles of choice, with slots indicating the affordances of each (e.g. one *stabs* with a knife, one *drives* a Mercedes but *sails* on a yacht, and one wears a hat on one’s *head* but wears shoes on one’s *feet*). The NOC also associates a character’s typical activities with apt locations (e.g. one *shops for shoes* in a *shopping mall* but *devises evil schemes* in an *underground lair*), while the taxonomic categories of each character are also organized into a type hierarchy. Importantly, the NOC is designed to be shared and modified in an open and cumulative fashion. It can be downloaded from *GitHub* as a convenient set of spreadsheets, containing approx. 30,000 semantic triples (see <https://github.com/prosecconetwork/>).

The NOC is sufficiently detailed to offer diverse points of overlap for a broad range of entities, real or fictional and historical or modern. Given any NOC character as a target of analysis, a system can quickly find potential sources of comparison amongst those that share a minimum number of overlapping qualities, such as in *domain* or *positive* and *negative* talking points. Or, as was the case with *Tarzan & IBM*, one can seek out antithetical entities that differ by virtue of opposing qualities (such as *boring & entertaining*, *smug & modest*, etc.) With an apt comparison or contrast in hand, generation then becomes a question of how best to frame the juxtaposition for rhetorical effect. The following rhetorical questions were generated by an autonomous bot named *@MetaphorMagnet* (Veale, 2015b) to squeeze into the 140-character message limit imposed by Twitter’s API:

What if [#TheEmpireStrikesBack](#) were real? [#HillaryClinton](#) could be its [#PrincessLeia](#): driven yet bossy, and controversial too

What if [#TheNewTestament](#) was about [#AmericanPolitics](#)? [#MonicaLewinsky](#) could be its [#Lucifer](#): seductive yet power-hungry, and ruined too.

Like Godard’s first choice of *Tarzan & IBM*, each of these pairings has one foot in reality and another in pure fantasy. In addition to the aptness of the pairings, it is this playful blurring of lines that contributes to the wit of each tweet. Indeed, it is the aptness of the comparison that justifies the blurring of the otherwise important boundary between fact and fiction. But one can go further than to say that Hillary and Leia, or Monica and Lucifer, are apt comparisons for one another, and suggest that they are also apt antagonists. After all, compelling stories are often built around conflict, and this conflict is most satisfying when it arises from the opposition of a well-matched protagonist and antagonist. What better way to ensure that two characters are different enough to be mutually antagonistic but similar enough to be well-matched than to require metaphorical equivalence? A story that pairs *Steve Jobs* and *Leonardo da Vinci* may be historically daft but it makes deft figurative sense, since each holds a comparable place in the public imagination: e.g., each is pioneering, artistic and far-seeing. The NOC allows one to find pairings that simultaneously make sense *and* nonsense, for stories to make readers think *and* smile.

Into the ‘Woulds’

Our conversations about narrative often resort to metaphor (see e.g. McKee, 2010). Yorke (2013) views story-telling as a walk into a mysterious woods, while in keeping with Campbell’s finding that heroic narratives often instantiate the *journey* schema, our most popular story metaphors treat narratives as having many of the properties of physical trajectories, such as pace and direction. We talk of plots as though they really could contain sudden twists and devious turns, of characters that go off the rails, of meandering tales that seem to go nowhere, and of tense, fast-paced stories that hurtle to nail-biting conclusions. As such, it is intuitive to think of plot as the track on which characters move forward or back, cross paths or occasionally collide. This track must be lain by narrative’s author, of course, so that characters – and audience members, as fellow travelers – may move along it as the shape of the story dictates. The job of an automatic story-teller then is to lay tracks that can take its characters from a chosen start-state to a fitting end-state and so describe the journey as to reward our interest.

Scalextric model race-tracks aim to capture the drama of a real car race at toy scale, by providing kids with diverse track segments – at 1:500 scale – to build complicated and perhaps even treacherous tracks of long straights, hair-pin bends and tricky chicanes. We aim to do much the same for story-telling systems that use *Scéalextric* for their plots, by providing a diverse array of clickable plot segments. As in *Plotto*, we assume that each segment has three clauses or actions, making each segment an action 3-gram. Consider, for instance, a plot segment comprising this action 3-gram:

A flatter B; B promote A; A disappoint B

A and B are placeholders that will be filled with specific protagonist & antagonist characters at the rendering stage. As for why A *flatters* B, this must be motivated by another segment that is clicked into place before this one, such as:

A read about B; B impress A; A flatter B

Likewise, what happens after A *disappoints* B may be this:

A disappoint B; B humiliate A; A attack B

Notice how each segment connects to others via a sharing of the first or last action, so the above three segments can be linked together, without repetition, to yield 7 scenes:

A read about B; B impress A; A flatter B; B promote A; A disappoint B; B humiliate A; A attack B

Diversity of generation is ensured when a multitude of plot segments, of three actions apiece, are available to a system. At present, *Scéalextric* provides 3000+ plot segments such as the above, collectively using over 800 action verbs. The collected segments form the equivalent of a textual n-gram language model whose outputs are plots, not sentences. By threading segments into a plot graph, a generator need only take a random walk in the graph to generate a logical plot. Before we consider how a system chooses a coherent start and end point for its walk, or how it determines the shape

of a resulting plot, we first consider how a plot is rendered.

The above three-segment plot of 7 actions is satisfying in its way, yet it is just a skeleton, not a rendered story. To give these skeletons an idiomatic surface form, we provide one or more idiomatic templates for each of the 800+ verbs in the teller’s repertoire. For instance, for *attack* we define:

A attacked B with all its strength. A launched a massive attack on B. A pounced on B. A threw itself into an attack on B. A launched a full-frontal attack on B. A clobbered B.

A mapping from skeletal plot to rendered narrative can be created by choosing randomly from the available idiomatic templates for each of the action verbs in the plot skeleton. We employ a similar mapping from plot verbs to pre-built prologue and epilogue texts. Any action verb that can serve as the first action of a narrative is linked to one or more opening texts, while any verb that can serve as the last action in a narrative is linked to one of more closing texts. For instance, these are the available prologues for *flatter*:

B’s ego thrived on flattery. B’s ego was a balloon inflated by flattery. B liked to be showered with flattery. B liked to be basted in the compliments of others.

A system can choose from the following epilogues for *kill*:

Well that was one way to deal with B! Well B won’t be bouncing back from that in a hurry! So A extracts a biblical justice from B. In effect A went medieval on B’s ass.

It follows that a system can begin a story at any plot action for which it possesses at least one prologue text, and end a story at any action that possesses at least one epilogue text. It is not the case then that the system first determines the starting and ending actions of a story and finds a sequence of other actions to coherently link them into a plot. Rather, if it provides apt prologue and epilogue texts for every verb in its inventory (which *Scéalextric* does, for all 800+ verbs) any linked sequence of plot n-grams will yield a valid plot.

We now turn the question of the causal connectives that connect successive actions, since the *twistiness* of a plot – which corresponds to the shape of a Scalextric track – is a function of how actions follow or defy causal expectations. The action B *promotes* A follows rather naturally from the prior action A *flatters* B since the goal of flattery is social gain, but the action A *disappoint* B is somewhat surprising once promotion is gained. When these are rendered we can thus expect a “so” connective to link A *flatter* B and B *promote* A and a “but” connective to link B *promote* A and A *disappoint* B. If inappropriate connectives are used a narrative will read as incoherent and confused, while if no connectives are used it may seem linear and uninteresting. Every *Scéalextric* 3-gram thus specifies an apt connective to link the first action to the second and link the second action to the third, in ways that respect a human reader’s causal intuitions. A good story will be neither too linear – too many *so*’s – nor too twisty – too many *but*’s – but will contain a balance of both. Moreover, when two *but*’s are used in sequence, the second is rendered as *yet*; when two *so*’s are used in sequence, the second is rendered as *then*.

Tying It All Together With Metaphor

The rendering process is complete once the system chooses characters to fill the *A* & *B* slots in its idiomatic templates. A generic Aesop-inspired strategy can simply choose two random story-book animals to fill the roles of *A* & *B*, such as a *bear* and an *eagle*, a *dog* and a *cat*, or a *rat* and a *crow*. The following fully-rendered *Scéalextric* story employs all of the aforementioned steps and resources before choosing two animals at random, a *snake* and a *koala*, for its *A* & *B*:

0. If anyone was in need of supervision it was the *koala*.
1. So at first, the vigilant *snake* supervised the juvenile *koala's* every activity.
2. But the *koala* could not reach the bar set by the *snake*.
3. So the *snake* considered the *koala* a loser.
4. Then the *snake* brutally beat the *koala*.
5. So the *koala* attacked the *snake* with all its strength.
6. But the *snake's* trickery went unnoticed by the *koala*.
7. So the *snake* decorated the walls with the *koala's* innards.
8. Then the *koala* assiduously curried favor with the *snake*.
9. But in the end the vigilant *snake* turned the juvenile *koala* into an indentured slave.
10. Thereafter the *koala* would wear the chains of a slave, but dreamt of choking the *snake* with those chains.

This 11-scene story is rendered from a plot skeleton of four connected action 3-grams – three actions apiece, with three shared overlapping verbs to connect them – that ekes out this chain of actions, with an added prologue and epilogue:

•—*are supervised by*—but→*fail to deliver for*—so→
→*disappoint*—so→*are beaten by*—so→*attack*—but→
→*are tricked by*—so→*are defeated by*—so→
→*curry favor with* —but→*are enslaved by*—•

As numbered above, scene 0 contains a fitting prologue for the opening action *are supervised by*, while scene 10 is an epilogue associated with the final action *are enslaved by*. Meehan's TALE-SPIN (1981) also used anthropomorphic animal fillers in the Aesopian tradition, yet such child-like conceits are suggestive of a toy world and a toy AI system. To open *Scéalextric* to the world of familiar human faces, we must integrate the NOC into the rendering process, not just by drawing *A* & *B* from the NOC but by integrating the affordances of the chosen characters into the rendering of actions, to add non-generic touches to an apt *mise en scène*. As noted previously, characters are paired on metaphorical grounds, so that *A* & *B* reside in different domains, genres or periods yet exhibit strong similarities, as is the case for *Jobs* & *Leonardo*, *Mahatma Gandhi* & *Obiwan Kenobi* or *Cicero* & *Obama*. Post-modern irony may also be used to assess figurative similarity via the NOC, so that *Lex Luthor*

and *Keyser Söze* are similar by virtue of having actor *Kevin Spacey* portray them both, while *Jor-El* and *Don Corleone* are similar because each was portrayed by Marlon Brando. Conversely, *Christian Bale* and *George Clooney* and *Adam West* are all well-matched as each has portrayed *Batman*.

A well-matched pairing of NOC characters should not be yoked to a plot with a random starting point; rather, the starting action – or at least one action in the initial segment – should befit the semantic types of the two characters. So if character *A* is a businessman but character *B* is a scientist we might expect *A* to *invest in* *B* or *B* to *impress* *A*. If *A* is a businessman and *B* is a reporter, then *B* may *interview* *A*. We thus provide several thousand story seeds that link two semantic types, as given in the NOC, with a starting action. When the system then seeks a plot to connect its chosen *A* & *B*, it picks a matching story seed and then seeks any plot in which the seed action is found in the first plot segment. The following *Scéalextric* story pits *Frank Underwood* (of TV's *House of Cards*) against the very real *Richard Nixon*. Since Frank and Richard are both politicians, one story seed that links them both has the action *campaign against*:

0. Richard Nixon and Frank Underwood were driven by very different political agendas.
1. So at first, Frank campaigned vigorously against Richard.
2. But Richard humiliated Frank by calling the sociopathic and ruthless Frank the Keyser Söze of wielding political power.
3. So the vindictive Frank hated Richard for being jowly, deceitful and secretive.
4. But Richard made a heartfelt appeal to Frank.
5. Then Frank's heart softened towards Richard.
6. So Frank forged a bond with Richard.
7. Then, Frank loyally sided with Richard in his struggles with John F. Kennedy.
8. But Richard underpaid Frank for his efforts plotting election strategies
9. So in the end the ruthless Frank cheated Richard out of a lot of money.
10. But those who cheat others have one fatal flaw: narcissism; it will be Frank's undoing.

Once again an 11-scene story is generated from a prologue, an epilogue and four plot segments of three actions apiece, where three duplicate actions overlap at the segment joins. The unrendered *Scéalextric* plot skeleton is as follows:

•—*campaign against* —but→ *are humiliated by* —so→
hate —but→ *are appealed to by* —so→ *are moved by*
—so→ *identify with* —so→ *show loyalty to* —but→
are underpaid by —so→*cheat*—•

Notice the integration of specific NOC affordances in the renderings of 3, 7 and 8. In scene 3 Underwood’s contempt for Nixon is vividly rendered with an appeal to the latter’s negative talking points. Scene 7, A *shows loyalty* to B, is given a character-specific rendering that shows familiarity with the history of *Richard Nixon*, while the rendering of scene 8, A *is underpaid* by B, is made specific to *Frank Underwood* by alluding to a typical activity from the NOC. When a rendering anchored in NOC details is available, it is always preferred to a generic idiomatic rendering.

Notice also the NOC-based rendering of scene 2, which does much more than simply integrate specific affordances from the NOC. Rather, following the Hollywood maxim of “*show, don’t tell*” this scene employs a novel speech act that is created on the fly to drive the plot forward. So the plot action A *is humiliated* by B is rendered not as a simple declaration that *Nixon* humiliates *Underwood* (a “tell”) but as a speech act that explains how *Underwood* is humiliated (a “show”). The renderer creates these speech acts as they are needed, using the very same capacity for metaphor that allows it to pair *Nixon* with *Underwood* in the first place. But the speech act in scene 2 is not just metaphorical; it is also wryly ironic in its breaking of the “fourth wall.” By comparing *Underwood* to the villainous *Keysar Söze* for apt story reasons, the renderer seems to be winking at the reader, for Nixon appears to know that actor Kevin Spacey portrayed both *Frank Underwood* in *House of Cards* and – spoiler alert! – *Keysar Söze* in *The Usual Suspects*. This is knowledge of the larger world that the NOC provides to the renderer, and as such it is grist for any metaphors and speech acts the renderer may need to generate as needed.

Empirical Evaluation

Two related approaches to story-generation were presented in the previous section. The first uses *Scéalextric* to build its plots around two randomly chosen Aesop-like animals, and renders this sequence of plot actions with a generic set of idiomatic mappings. The *koala* and *snake* narrative of the previous section exemplifies this baseline approach. The second approach uses the same plotting mechanism – clicking together overlapping segments to form a coherent whole – but uses the NOC to choose *human* characters, to choose the initial plot segment to suit the semantic types of these characters, and to opportunistically render individual plot actions using familiar associations from the NOC. The *Nixon vs. Underwood* story of the last section exemplifies this NOC-based, character-led approach. So each approach uses precisely the same plotting mechanism but employs different means to render its plots as polished surface texts.

We predict that plots rendered with familiar details from the biographies of its characters will be perceived as more vivid, more entertaining and even more dramatic than stories rendered using only generic surface forms. To test this prediction we generate 50 rendered instances of each kind of story and elicit ratings for each from human judges

on the crowd-sourcing platform *CrowdFlower.com*. Each instance is generated following the mechanisms described in the previous section, while the choice of which instances are shown to the judges is randomly determined. Although many factors influence a reader’s enjoyment of a narrative – for example, whether an odious character gets his comeuppance, or whether a virtuous character finds her reward – we expect that these factors will balance themselves out in a random sampling of all the stories that can be generated.

Judges were not informed as to the mechanical provenance of the experimental stimuli, but were simply told that each story was harvested from Twitter. 10 ratings were sought for 6 dimensions for each of the 50 NOC-based stories and 50 generic stories: *laughter* (how likely is this story to make someone laugh?), *entertainment* (how entertaining is this story?), *imagination* (does this story show evidence of an active imagination?), *vividness* (how memorable are the elements of this story?), *silliness* (how implausible is this story?) and *drama* (how eventful is this story?). Judges were shown just one story at a time and asked to rate just one dimension of each, on a scale of 1 (low) to 5 (high). A pool of judges was provided by CrowdFlower, allowing 10 ratings per stimulus to be averaged. We did not require all judges to rate all stimuli, so we report no measures of inter-annotator agreement. Note also that judges were not asked to directly rate the “creativity” of any stimulus, as notions of what constitutes creativity, and how to elicit numbers for those notions, vary significantly (see Jordanous, 2012).

So 10 independent human ratings were elicited for the 6 dimensions of the 50 NOC-based stories and the 50 generic stories, as outlined above. Judges were paid a small sum for each of their ratings, and scammers were eliminated in the usual way: a question requiring a non-random answer (e.g. “how many letters are in the word <W>”) was used to separate engaged users from disengaged cheats. To ensure that ratings for one dimension of a story did not influence a judge’s ratings for other dimensions, judges were presented with just one story – and asked to rate just one dimension of that story – at a time. As shown in Veale & Alnajjar (2016), eliciting multiple ratings for the same stimulus in the same task unit can cause interference in the results, causing one dimension to influence a judge’s rating of another. The mean ratings across stories and judges for 6 dimensions of each story type (and *All* taken together) are shown in Table 1. (Standard deviations are in parentheses.)

Table 1. Mean ratings per dimension for each type of story

Dimension	NOC-based	Generic	All
Laughter	3.02 (1.13)	2.33 (1.13)	2.67
Entertainment	3.10 (1.15)	2.91 (1.12)	3.00
Imagination	3.43 (1.04)	3.03 (1.08)	3.23
Vividness	3.48 (0.97)	2.90 (1.05)	3.19
Silliness	3.47 (1.04)	2.98 (1.2)	3.22
Drama	3.53 (0.98)	2.93 (1.14)	3.23

So we see significant improvements across all dimensions of evaluation, with the stories generated from metaphoric pairings of NOC characters that integrate familiar aspects of those characters into their renderings outperforming the stories that rely on generic characters and renderings. After Bonferroni correction is applied the improvements (*NOC* over *Generic*) remain significant at the $p < .001$ level for all dimensions except *Entertainment*, for which $p < .01$.

The most dramatic improvement can be seen, fittingly, in the dimension *Drama*. Though NOC-based and generic stories are each rendered upon a plot skeleton that is built in precisely the same way – save for the added caveat that NOC stories require an action in the initial plot segment to reflect the semantic types of the characters involved – the use of familiar characters and their vivid associations lend the actions of the plot a comic and exaggerated quality that appears to enhance the perceived eventfulness of the story. The prior expectations that readers bring to the NOC-based stories appear, in the main, to make actions and plot turns more engaging than when the same actions and turns are woven around the ephemeral characters of a more generic tale. As a finding about story-telling this is very old news, for scholars since Aristotle have recognized the importance of integrating character *and* plot to build a satisfying tale. It is nonetheless a welcome finding in an empirical context that brings complementary large-scale resources together for the purpose of automatically generating more engaging and entertaining tales on an industrial scale. The NOC list, which supports the automated creation of vivid metaphors, and *Scéalextric*, which turns the creation of story plots into a random walk in a structured forest of causal possibilities, work well together as a generator of interesting stories that achieve a more perfect union of character *and* plot.

Beyond Textuality: Multi-Modal Renderings

Reiter & Dale (2006) note that the generation of complex natural-language artefacts requires two levels of planning: *macro*-planning (what is it that I want to say?) and *micro*-planning (so how do I go about saying it?). This division of levels is found in our separation of plot generation and the subsequent rendering of this skeletal plot, in which actions are mapped to surface-level idiomatic forms. By defining more idiomatic templates for the 800+ verbs that make up *Scéalextric*'s plot segments, we provide a greater flexibility in rendering, allowing a story-generator to render its plots in more varied ways that read as fresh and unmechanical. To generate stories in German, French, Spanish or Klingon we need only provide the corresponding stock of idiomatic templates for the action verbs of the *Scéalextric* generator. But those alternate idiomatic templates are not restricted to textual encodings of spoken language, and may incorporate – or rely entirely upon – pictorial elements such as *Emoji*.

We can replace animal designators such as *koala* and *dog* with their corresponding Unicode characters when rendering the *A* & *B* fillers of generic story-lines, making our inventory of story animals co-extensive with that of

animal *Emoji* in the Unicode standard. In this rendering of the action *are bought off by*, in which *B* is a snake and *A* is a pig, each character is easily replaced with an *Emoji*:

Then the 🐍 bribed the 🐷 to play along.

In addition to providing textual idiomatic forms for each of the system's 800+ action verbs, we can also provide *Emoji* translations for each verb. The *rebus* principle allow us to use *Emoji* as both *pictograms* (images depicting ideas) and *sound images* (images depicting words that imply sounds) so that the above scene can be rendered entirely in *Emoji*:

🐷🗣️🐍👉🐷🙌🚫🐷🙌🐷 (or: 🐷 was bought off by 🐍)

Then the 🐍 bribed the 🐷 to play along.

An *Emoji* mapping for each of *Scéalextric*'s 800+ verbs is engineered manually, as this task requires some ingenuity. As shown above, a full *Emoji* rendering is presented side-by-side with its comparable text rendering (and a linguistic short-hand in parentheses), to allow readers to familiarize themselves with this new visual idiom at their own pace. Pure *Emoji* offers remarkable concision, allowing an entire story to be summarized in a single picture-only tweet:



Emoji serve largely at present as visual adornments for our textual renderings, rather like cute story-book illustrations. Though *Emoji* are not pictograms in a strong sense – they are far too ambiguous to serve reliably in this role – they nonetheless constitute a lexicon of visual ideas that reflects the collected interests of contemporary social media users. We plan to further explore the role of *Emoji* as proxies for the semantic primitives that comprise the semantic lexicon of a story-telling system, to achieve a stronger integration of plot, character *and* mental image in the tales that we tell.

Conclusions: Once More Unto The Breach

Scéalextric and the *NOC list* were each designed with the express purpose of supporting research in computational creativity that is practical, scalable *and* knowledge-driven. For each owes its genesis to the international student code-camps for which it was first created and from which each

has later grown in scale and complexity. Researchers may access either resource (and related code) in a public Github that is frequently updated: github.com/proseconetwork

But *Scéalextric* and the NOC must grow and evolve to remain relevant as comprehensive resources for research. For the NOC this means the inclusion of new cultural figures as they reach iconic status, while for *Scéalextric* this means tackling the various weaknesses of the *plot-as-path* approach as it now stands. For instance, the plotting mechanism currently assumes that each story has just two characters who move through a tale in parallel, whereas Campbell and Propp allow for a retinue of other characters to participate in the action. To address this shortcoming we will take a leaf from Plotto, which assumes that additional characters can be functions of the protagonist *A* (such as *F-A*, *father of A*) or antagonist *B* (such as *S-B*, *spouse of B*). In this way the supporting figures can be woven into the action as they are needed. As *Scéalextric* graduates from juggling *two* balls to juggling *many* at once, it can graduate to telling nuanced stories about real (or at least familiar) human characters of near-human-level complexity.

Acknowledgements

This work was conducted as part of the EC-funded projects PROSECCO and WHIM. The experimental results were obtained in collaboration with Alessandro Valitutti. The Emoji verb-mapping was constructed by Philipp Wicke.

References

- Brants, T. and Franz, A. 2006. Web 1T 5-gram Version 1. *Linguistic Data Consortium*.
- Burroughs, W. S. (1963). *The Cut Up Method*. LeRoi Jones (ed.), *The Moderns: An Anthology of New Writing in America*. NY: Corinth Books.
- Campbell, Joseph. *The Hero with a Thousand Faces*. Princeton: Princeton University Press.
- Cook, William Wallace. (1928/1981). *William Wallace Cook. Plotto: The Master Book of All Plots*. Reprinted by Tin House Books (isbn: 9781935639183).
- Fauconnier, G. & Turner, M. (1998). Conceptual Integration Networks. *Cognitive Science*, 22(2):133–187.
- Fauconnier, G. & Turner, M. (2002). *The Way We Think. Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.
- Gervás, P. (2013). Propp's Morphology of the Folk Tale as a Grammar for Generation. In *Proceedings of the 2013 Workshop on Computational Models of Narrative*, Dagstuhl, Germany.
- Gervás, P., Hervás, R., León, C. & Gale, C.V. (2016). Annotating Musical Theatre Plots on Narrative Structure and Emotional Content. In *Proc. of the 7th International Workshop on Computational Models of Narrative*, Krakow, Poland.
- Giora, R. (1997). Understanding figurative and literal language: The graded salience hypothesis. *Cogn. Linguistics* 8-3:183-206.
- Giora, R., Fein, O., Ganzi, J., Levi, N.A. & Sabah, H. (2004). *Weapons of Mass Distraction: Optimal Innovation and Pleasure Ratings. Metaphor and Symbol* 19(2):115-141.
- Jordanous, A. (2012). A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4.3: 246-279.
- McKee, R. (2010). *Story: Style, Structure, Substance, and the Principles of Screenwriting*. New York: Harper-Collins.
- Meehan, J. (1981). *TALE-SPIN*. In Shank, R. C. and Riesbeck, C. K., (eds.), *Inside Computer Understanding: Five Programs plus Miniatures*. Hillsdale, NJ: Lawrence Erlbaum.
- O'Brien, F. (1939/1998). *At Swim Two Birds*. Dublin, Ireland: Dalkey Archive Press (1998 reissue).
- Pérez y Pérez, R. & Sharples, M. (2004). Three Computer-Based Models of Storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge Based Systems Journal*, 17(1):15-29.
- Propp, V. (1928/1968). *Morphology of the Folktale*. University of Texas Press (2nd edition; English translation by Laurence Scott).
- Reiter, E. & Dale, R. (2006). *Building Natural Language Generation Systems (Studies in Natural Language Processing)*. Cambridge University Press.
- Riedl, M. O. and Young, R. M. (2010). Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research* 39.1, 217-268.
- Schank, R. & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Psychology Press: New York, NY.
- Turner, S.R. (1994). *The Creative Process: A Computer Model of Storytelling*, Hillsdale, NJ: Lawrence Erlbaum.
- Veale, T. & Hao, Y. (2008). Talking Points in Metaphor: A Concise Usage-based Representation for Figurative Processing. In *Proc. of ECAI 2008, the 18th European Conference on Artificial Intelligence*. Patras, Greece.
- Veale, T. (2014). Coming Good and Breaking Bad: Generating Transformative Character Arcs For Use in Compelling Stories. In *Proceedings of ICC-2014, the 5th International Conference on Computational Creativity, Ljubljana, Slovenia, June 2014*.
- Veale, T. (2015a). Fighting Words and Antagonistic Worlds. In *the Proceedings of the 3rd Workshop on Metaphor in NLP at the North American ACL (NAACL)*, Denver, Colorado, June 5.
- Veale, T. (2015b). Unnatural Selection: Seeing Human Intelligence in Artificial Creations. *Journal of General Artificial Intelligence*, 6(1):5-20.
- Veale, T. & Alnajjar, K. (2016). Grounded for life: creative symbol-grounding for lexical invention. *Connection Science*, 28(2):139-154.
- Veale, T. (2016). A Rap on the Knuckles and a Twist in the Tale: From Tweeting Affective Metaphors to Generating Stories with a Moral. In *Proceedings of the AAAI Spring Symposium on Ethical and Moral Considerations in Non-Human Agents*
- Vogler, S. (1984/1998). *The Writer's Journey: Mythic Structure For Writers*. Studio City, CA: Michael Wiese Productions (a book treatment of Vogler's original 7-page memo from 1984).
- Yorke J. (2013). *Into the Woods: A Five-Act Journey into Story*. London, UK: Penguin.

How to Build a CC System

Dan Ventura

Computer Science Department
Brigham Young University
Provo, UT 84602 USA
ventura@cs.byu.edu

Abstract

Building a computationally creative system is a challenging undertaking. While such systems are beginning to proliferate, and a good number of them have been reasonably well-documented, it may seem, especially to newcomers to the field, that each system is a bespoke design that bears little chance of revealing any general knowledge about CC system building. This paper seeks to dispel this concern by presenting an abstract CC system description, or, in other words a practical, general approach for constructing CC systems.

Introduction

The broad field of computational creativity (CC) admits a range of autonomy, from creativity support tools to co-creative systems to fully autonomous artificial agents, and it is this last extreme which is the focus here. The notion of an (autonomous) creative agent has been instantiated in many different forms, and a variety of systems of varying degrees of sophistication and efficacy have been built by the CC community for creating artefacts in a broad range of domains. Many of these systems have been documented in the literature and their mechanisms described in some level of detail. The goal of this paper is to generalize multiple approaches from different domains into an abstract system description that provides a sort of blueprint for how to build a CC system for an arbitrary domain. The intent is to provide a fairly straightforward distillation of (one view of) what is involved in the construction of such a system, both to provide a hands-on guide for the newcomer wishing to build such a system and to stimulate discussion about what exactly *is* the right way to go about building such a system.

To begin with, in this paper, *computationally creative agent* means an agent whose behavior exhibits three characteristics: *novelty*, *value* and *intentionality*. Note that the first two are most commonly addressed with respect to product¹, while the third deals with process. For the purposes of this discussion, these characteristics will be defined as follows:

novelty: the quality of being new, original or unusual; this is relative to the population of artefacts in the domain in question and can apply in the personal or historical sense.

¹But also note that the use of “product” here is abstract, and that, in particular, the artefact produced might itself be a process.

value: the importance, worth or usefulness of something; this would typically be ascribed by practitioners of the domain in question.

intentionality: the fact of being deliberative or purposive; that is, the output of the system is the result of the system having a goal or objective—the system’s product is correlated with its process.

The goal here is to lay out how to build an autonomous CC system, but it is, of course, possible to selectively apply parts of what follows to build co-creative systems or even “simple” creativity support tools as well, with the differentiation (most) dependent on the level of creative responsibility with which the final system is endowed. It is important to note that the guidelines presented here are one current view of things, based on experience both building and reviewing multiple CC systems in widely differing domains, and while the general concepts are meant to be somewhat definitive, the examples and methodological suggestions given are meant to be representative rather than prescriptive.

Building the System

The goal of an autonomous CC system is to intentionally produce artefacts that are both novel and valuable in a given domain. Figure 1 gives a functional design for such a system, embedded in the target *domain*. The system is embedded in the domain for several reasons: it has a domain-specific *knowledge base*; it has a domain-appropriate *aesthetic*; and it has the ability to externalize artefacts that potentially can contribute to the domain. In addition, the system has an internal *representation* of artefacts that allows it to reason about domain-related concepts and manipulate these concepts to *generate* potential artefacts. It also has the ability, based on its aesthetic, to *evaluate* both its internal *conceptualization* and the *translation* of this conceptualization into the realization of an artefact in the domain. Each of these components will be considered in turn, while emphasizing that there is not a strict linear ordering to either their development or their deployment; rather, both system development and system operation are more likely semi-ordered, iterative processes.

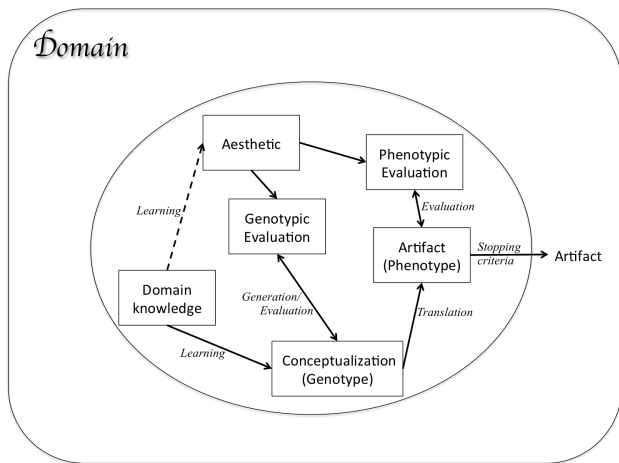


Figure 1: An abstract CC system is embedded in a particular domain of interest by incorporating both a repository of domain knowledge and a domain-appropriate aesthetic that together inform the production of artefacts that (potentially) contribute to that domain. Artefacts are represented internally as a genotypic conceptualization that is manipulated and evaluated internally and is eventually translated into an externalizable phenotypic representation that is further evaluated for its suitability before (potentially) being exported to the domain.

Domain

In contrast to traditional artificial intelligence tasks, which are most often characterized with an objective function that is to be maximized (or minimized), the kinds of problems that CC systems are built to solve are of an entirely different class. There is no such thing as a best song, or best theorem or best design. One cannot maximize a piece of visual art or a recipe or a poem. There are many interesting songs, theorems, designs, paintings, recipes and poems, and the goal is to find one or more of these. What constitutes a “solution” for these types of “problems” is nothing like an optimization problem, at least in the traditional sense. The first step in building a CC system is to choose a domain D for which it would be useful to build such a system. Because almost any domain of endeavor can be argued to require creativity to meet at least some of its challenges, from the artistic to the scientific to the mundane, the choice is really limited only by the imagination. Indeed, successful CC systems already exist for a large variety of domains, including culinary recipes (Morris et al. 2012; Varshney et al. 2013), language constructs such as metaphor (Veale and Hao 2007) and neologism (Smith, Hintze, and Ventura 2014), visual art (Norton, Heath, and Ventura 2013; Colton 2012), poetry (Toivonen et al. 2012; Oliveira 2012; Veale 2013), humor (Stock and Strapparava 2003; Binsted and Ritchie 1994), advertising and slogans (Strapparava, Valitutti, and Stock 2007; Özbal, Pighin, and Strapparava 2013), narrative and story telling (Riedl and Young 2010; Pérez y Pérez and Sharples 2004), mathematics (Colton, Bundy, and Walsh 1999), games (Cook, Colton, and Gow 2016; Liapis, Yannakakis,

and Togelius 2012) and music (Bickerman et al. 2010; Pachet and Roy 2014). Of course, none of these domains can yet be considered “solved” by CC (indeed, for CC problems, it is not clear that the idea of “solving” even makes sense), so much work remains to be done even here. However, certainly there are many more domains for which the development of CC systems will prove beneficial: product design (a very general domain that could be further specialized to automobile design, electronics, clothing, software apps, etc.), architecture, drug design, protein synthesis, trip planning, robotics (physical systems, path planning, goal generation, etc.)—the list is endless.

Representation

Given a domain, it is necessary to next choose an appropriate representation for artefacts in that domain, and for the general case, that entails actually choosing two representations: a *phenotype* p that is an external, public representation, and a *genotype* g that is an internal, private representation.²

Given the domain D , the phenotypic representation is often at least somewhat prescribed. For example, for the domain of narrative, the phenotype must be some version of a story, for the domain of music it must be some kind of song, for visual art, a painting, for mathematics, a theorem. However, it may be convenient to choose some modification $P \simeq D$ as the phenotypic representation: a story outline, a lead sheet, a digital image, a sequence of predicates. Then, a specific phenotype p is the representation of an artefact in the domain D (or its surrogate P). So, $p \in P \simeq D$.

The genotype representation G may be very different from both D and P ; it should be a convenient form for knowledge representation, reasoning, and manipulation. Using the same four examples, for the domain of narrative, it might be entity-relationship graphs, schemata or plans; for music, it might be MIDI or lead sheets or viewpoints; for visual art, it could be arrays of pixel values, sequences of image filters or sets of image segments; for mathematics, it might be Prolog programs, trees or Boolean formulas. A specific genotype $g \in G$ is an encoding of a phenotype p that is used internally by the system.

It will also be important to decide on a representation for domain knowledge that will be used as the system’s knowledge-base or experience. It may be convenient to adopt the genotypic representation for representing knowledge about the domain, or, it may be convenient to use the phenotypic representation, or both.

Knowledge Base

Having the question of domain knowledge representation settled, the next task is deciding on a way to collect representations of that knowledge into a knowledge base K . This will serve as the system’s experience and provide it with its connection to the domain. Perhaps the most common means of populating K is by leveraging the web in some way: scraping websites and cleaning the obtained data,

²We appropriate these terms without intending to imply the normal biological or evolutionary connotations with which they are usually associated.

open-access or for-purchase corpora or databases, crowdsourcing, etc. K can also be populated using experts to construct bespoke rules, prototypes, knowledge graphs, semantic networks and the like. In the case of a system for creating recipes, one might scrape recipe websites for ingredients, example recipes, their rankings, their categories, etc.; for a poetry creation system, resources such as Google n-grams, WordNet and ConceptNet might serve as a knowledge base; for a joke-writing system, the knowledge base might contain a set of rules constructed by professional humorists; for a system intended to invent board games, K could consist of a set of known board games (represented in an appropriate language, such as the Stanford GDL (Love et al. 2006)). This knowledge base K is used as a starting point for the rest of the system; in particular, it will be used to learn some kind of conceptual model of the domain and may also be used to learn an appropriate aesthetic as well.

Aesthetic

An aesthetic A is an abstract measure of quality for artefacts in the domain. Given the goal of producing artefacts that are valuable and novel, this quality should in some way be correlated with these. Continuing the second set of examples, aesthetic considerations for recipes might include

- appeal (is it tasty?),
- nutritional value (is it healthy?),
- cost/availability of ingredients (is it affordable?),
- surprising flavors (do these flavors somehow complement each other?), etc.;

for poetry, they might include

- semantic coherence (does it make sense?),
- interestingness of theme (will it hold the reader's attention?),
- metrical and/or rhyming considerations (is it interesting to read?)
- and cultural reference (does it apply?);

for jokes, good measures may include

- funniness (will it make people laugh?),
- accessibility (will people get it?),
- surprise (is it different than expected?),
- timeliness (does it reference pop culture or current events?),
- potential for shock or insult (will it make people angry?);

for board games, aesthetic factors might include

- playability (do the rules actually work?),
- winnability (can the game be won?),
- amount of time required (is it too long?),
- complexity (will people understand how to play?),
- enjoyability (is it fun to play?),
- and social considerations (how many people play? how do they interact?).

The most straightforward way to imbue the system with an aesthetic is to simply give it one—as the system designer, decide on some set of measures for the system to use.

Another approach is to have the system learn an aesthetic from the knowledge base, $A = \lambda_a(K)$. That is, given the information in K about (presumably both good and bad) artefacts from the domain, the system makes use of some learning function $\lambda_a : \mathcal{K} \rightarrow \mathcal{A}$, to infer an aesthetic $A \in \mathcal{A}$, where \mathcal{K} is the set of all knowledge bases and \mathcal{A} is the set of all aesthetic measures. This is appealing for its greater system autonomy, but it may be difficult to effect, and the result may not be interpretable (i.e., one may not know what aesthetic the system is using, though some will argue this as a positive advance).

Conceptualization

A conceptualization C of K is some kind of model of the knowledge that facilitates the understanding of, and thus the creation of, artefacts in the domain D (or the surrogate domain P). This model is constructed via some kind of learning process $\lambda_c : \mathcal{K} \rightarrow \mathcal{C}$, where \mathcal{C} is the set of all conceptual models; so, $C = \lambda_c(K)$. The form of this conceptualization can vary widely, but ideally it should admit some method of reasoning about it and should be mutable. It should also facilitate generation of artefact genotypes. Representative examples include, for narrative, sets of characters, relationships and actions or an engagement-reflection model; for music, (hidden) Markov chains of transition probabilities between pitches, durations or chords or probabilistic context sensitive grammars; for visual art, a library of semantically clustered images or generative adversarial networks; for mathematics, axioms and operators or genetic programs; for recipes, a list of ingredients and their relative or absolute statistics or a model of chemical properties of ingredients; for poetry, templates or recurrent neural networks or n-grams; for jokes, templates or skip-thought vectors; for board games, a probabilistic grammar or set of cases.

Generation

The conceptualization should allow the system to generate artefact genotypes via some generative function $\gamma : \mathcal{C} \times \mathcal{S} \rightarrow G$, where \mathcal{S} is a placeholder set meant to include anything that might be useful in the generation process: inspiration sources, randomness, examples, or even nothing at all. This generative process might be a natural extension of the conceptualization, or it might be quite distinct from it. Examples of the former include (hidden) Markov chains (just sample the chain for some length of sequence), probabilistic context sensitive grammars (sample a grammatical derivation), generative adversarial networks (cycle between generative network and adversarial network until output stabilizes/error is low), recurrent neural networks (just stimulate the network and record the output for the desired sequence length). Examples of the latter include combinatoric approaches, chaining pre- and post-conditions, logic programs, genetic algorithms, genetic programs, physical modeling or simulation, template filling, and nearest neighbor methods.

Genotypic Evaluator

Given the ability to generate candidate genotypes, the system requires some way to evaluate those genotypes—those judged to have quality will be converted to phenotypes that are further evaluated and may be released to the domain as successful creations; those judged unfit can be discarded or possibly modified to increase their quality. This evaluation should consist of both a domain-specific assessment of the value of an artefact and some kind of similarity measure for artefacts; these combined give a measure of both value and novelty—the value rule allows filtering for value potential and the similarity measure can be used to compare a candidate with the population of K as well as with artefacts the system itself has previously created. Thus, what is required is an evaluation function $\epsilon_g : G \rightarrow [0..1]$ which assigns some real-valued quality score to an artefact g by taking into account both the value and the novelty of G , perhaps most simply as a linear combination: $\epsilon_g(g) = \alpha v_g(g) + (1 - \alpha) \nu_g(g)$, where $v_g : G \rightarrow [0..1]$ computes a value score for g , $\nu_g : G \rightarrow [0..1]$ computes a novelty score for g and $0 \leq \alpha \leq 1$.

The novelty score returned by ν_g should correlate in some way with a notion of distance from known artefacts (i.e. those in $K \cup Z$, where Z contains those artefacts [already] successfully created by the system), with a higher novelty score indicating a greater distance from known artefacts. The notion of distance will be representation-specific, of course, and could be something as simple as a Hamming distance (in the case of binary strings), as common as Euclidean distance for representations as points in \mathcal{R}^n or as exotic as a generalized edit distance, where the notion of editing is appropriately defined. Though this distance will typically be explicitly designed based on the representation, there may be situations in which it could make sense for the system to learn a notion of distance (or even novelty directly) empirically from data in K .

The value score returned by v_g should correlate with the aesthetic A . That is, there should be some mapping $\phi_g : \mathcal{A} \rightarrow \mathcal{E}$, where \mathcal{E} is the set of valuation functions whose domain is G and whose range is $[0..1]$. This mapping will likely be something *ad hoc*, with the function v_g , being explicitly designed as part of the system, though again, it may be possible to instead implement ϕ_g and allow the system to learn v_g . Another way to think of this is that v_g operationalizes A . Examples of a valuation function might include measure of affect, or sentiment (via dictionaries) or tension/resolution for stories; measuring melodic shape, number of key changes, pitch range, uniqueness of chord progression for music; affect, subject matter, color usage, style for visual art; generality or simplicity for mathematics; existence of unique ingredient combinations, complementary chemical taste profiles, number of calories or cost of ingredients for recipes; identifiable meter, rhyming, affect, sentiment, word usage for poetry; word usage, current event usage, sentiment, incongruity in jokes; number of turns required, rule complexity, number of players, number of ludic conditions, likelihood of a draw for board games.

Finally, it should be mentioned that it is possible in some cases to partially or completely incorporate the functionality

of ϵ_g into the generative function γ , obviating the need for a separate evaluation of g . For example, if a certain metrical structure or rhyme scheme is highly valued in a poetic form, the generative process may constrain all outputs to follow that structure, or if the combination of dairy and meat products in a recipe were considered undesirable, the generative process may disallow the combination. Of course, this kind of constrained generation precludes exploring some parts of the domain, so it should be used cautiously.

Translation

Given that the system is working with an internal, genotypic representation g but that it must ultimately produce an external, phenotypic representation p , some method of translation is necessary. This translation mechanism may be thought of as a helper function $\tau : \{G \cup \perp\} \rightarrow \{P \cup \perp\}$, where \perp represents the null artefact, which produces a phenotype from a genotype; thus, $p = \tau(g)$ and $\perp = \tau(\perp)$. Using τ , the system turns a schema into a story or a Markov chain into sheet music or a sequence of image filters into an image or a Prolog program into a proof or a binary string into a list of ingredients or a template into a poem or a bag of words into a joke or a probabilistic grammar into a board game (description). This may be one of the easiest parts of the system (e.g., obtaining an image from a sequence of image filters is usually a simple matter of composing a few well-defined function calls) or one of the most difficult parts (e.g., composing a punchline from a bag of key words to make a joke funny is completely ill-defined).

Phenotype Evaluator

Genotypes that are evaluated highly enough will be translated (via τ) into candidate phenotypes that must be evaluated in their own right, with a function $\epsilon_p : P \rightarrow [0..1]$. This evaluation should be qualitatively different than its genotypic counterpart because a) this is a different representation and b) there is nothing to be gained by re-using the same evaluation criteria. This is analogous to evaluating a piece of sheet music (genotype) by checking its agreement with music theoretic principles on the one hand and listening to how an audio track (phenotype) of the music sounds when played (translation). This function, too, should correlate with notions of value and/or novelty. And, again, like ϵ_g , ϵ_p may be designed as an(other) operationalization of A , or it may be learned by the system using a function $\phi_p : \mathcal{A} \rightarrow \mathcal{E}$.

While the evaluator ϵ_g is most often somewhat piecewise and (in some sense) cognitive in nature, ϵ_p may be more holistic and very often is based in some form of perception. Because of this, domains seem to vary widely in the ease with which a phenotypic evaluator can be developed. For some domains, we have a fairly good understanding of the perception necessary to construct a phenotypic evaluation: some kind of audio signal processing for music; computer vision-based techniques for visual art; chemical analysis for recipes; (simulated) game play for board games. For other domains, this perceptual understanding is much less developed, and it is less clear how phenotypic evaluation can be done: how do readers perceive a story as interesting? how

do mathematicians see beauty in a theorem? how do listeners feel an emotional connection with a poem? how does an audience find a joke funny?

Putting it all Together

With all the pieces in place, it is now possible to summarize the entire process as follows. To build a CC system and produce an artefact, follow these steps:

1. Choose a domain D
2. Choose a genotypic representation G and a phenotypic representation $P \simeq D$
3. Collect data and build a knowledge base K
4. Choose a generator function γ
5. Choose aesthetic A [or build a function λ_a that allows the system to learn $A = \lambda_a(K)$]
6. Choose a novelty measure ν_g and a value measure v_g [or build a function ϕ_g that allows the system to learn $\nu_g = \phi_g(A)$] and construct a genotypic evaluator ϵ_g from ν_g and v_g
7. Choose a phenotypic evaluator ϵ_p [or build a function ϕ_p that allows the system to learn $\epsilon_p = \phi_p(A)$]
8. $C \leftarrow \lambda_c(K)$
9. While $\epsilon_p(p) < \theta_p$
 - While $\epsilon_g(g) < \theta_g$
 - $g \leftarrow \gamma(C, \rho)$
 - $p \leftarrow \tau(g)$
10. return(p)

Looking at this more formally requires two additional helper functions, $\Theta_g : \{G \cup \perp\} \rightarrow \{G \cup \perp\}$ and $\Theta_p : \{P \cup \perp\} \rightarrow \{P \cup \perp\}$, defined as follows:

$$\Theta_g(x) = \begin{cases} \perp & \text{if } x = \perp \\ x & \text{if } \epsilon_g(x) \geq \theta \\ \perp & \text{if } \epsilon_g(x) < \theta \end{cases}$$

where θ is some threshold of acceptability, and Θ_p is defined similarly. With these, the entire system operation for artefact creation can be expressed as

$$\begin{aligned} a &= \Theta_p(\tau(\Theta_g(\gamma(\lambda_c(K), \rho)))) \\ &= \Theta_p(\tau(\Theta_g(\gamma(C, \rho)))) \\ &= \Theta_p(\tau(\Theta_g(g))) \\ &= \Theta_p(\tau(g)) \\ &= \Theta_p(p) \\ &= p \end{aligned}$$

when $\Theta_g(g) = g$ and $\Theta_p(p) = p$, and $a = \perp$ otherwise.

Variations and Further Considerations

Simplifying

For some domains, it might be unnecessary to work with both phenotypic and genotypic representations. While, in

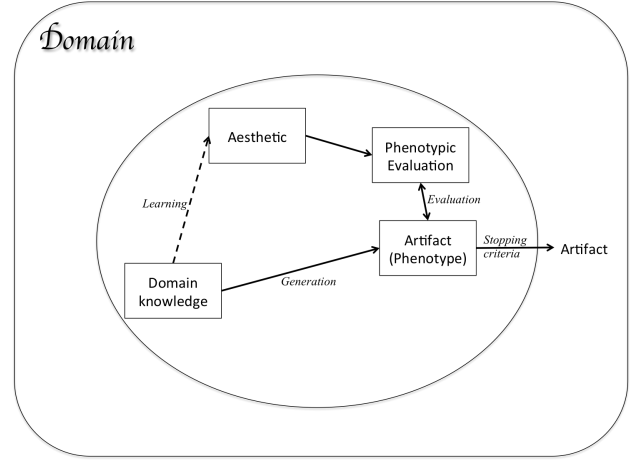


Figure 2: A simplified abstract CC system may eschew an internal representation, simplifying both the generation and evaluation processes. The system is still embedded in a domain with both a knowledge base and an aesthetic, but now its internal and external representations are equivalent, obviating the need for differentiated evaluation/generation mechanisms. It is likely that such a system will be limited in its ability to intentionally produce quality artefacts for most domains due to the lack of an ability for conceptualization.

general, this will likely limit the ability of the system to produce quality artefacts, in the case of simple tasks, prototyping or other less complex scenarios, it may be possible to work directly with only a phenotypic representation. This simplifies system design significantly (see Figure 2).

It is still necessary to choose a domain D and phenotypic representation $P \simeq D$, collect a knowledge base K , produce an aesthetic A and its operationalization in the form of a phenotypic evaluation function ϵ_p ; however, the list of components no longer required includes the genotypic representation G , the translation function τ , the genotypic evaluator ϵ_g , the conceptualization model C and the learning mechanism λ_c . In addition, because the system no longer has any conceptualization nor genotypic representation, the generation function γ must be modified so that it depends directly on K rather than on C and so that it outputs a phenotype p rather than a genotype g ; thus, now $\gamma : \mathcal{K} \times \mathcal{S} \rightarrow P$.

Then, the simplified process is

1. Choose a domain D
2. Choose a phenotypic representation $P \simeq D$
3. Collect data and build knowledge base K
4. Choose a generator function γ
5. Choose aesthetic A
6. Choose a phenotypic evaluator ϵ_p
7. While $\epsilon_p(p) < \theta_p$
 - $g \leftarrow \gamma(K, \rho)$
8. return(p)

and the formal description simplifies to

$$\begin{aligned} a &= \Theta_p(\gamma(K, \rho)) \\ &= \Theta_p(p) \\ &= p \end{aligned}$$

Complexifying

Of course, in many more situations, not only will it not be possible to simplify the original system as just discussed, but also it will likely be necessary to introduce further complexity to obtain satisfactory results. This complexification can come in many forms, and more of these will be discussed later, but the most natural next step is likely to introduce further domain-system interaction in the form of teaching and feedback. This ability allows the system to change over time, assimilating new domain knowledge as it becomes available and incorporating feedback about its creations. This new knowledge and feedback can directly affect the knowledge base K , directly or indirectly affect the aesthetic A and directly or indirectly affect the conceptualization C (see Figure 3). These effects can (and should) be propagated throughout the system, introducing the need for a time index t and obviating the need for some initial design decisions (e.g. even the knowledge base does not need to be fixed in advance). This facilitates dynamic knowledge acquisition, conceptualization, evaluation and generative abilities. Thus, the system can react to changes in the environment and become distanced from initial designer decisions.

Once again revisiting the process, a time index is now incorporated, but the overall flow remains recognizable:

1. Choose a domain D
2. Choose a genotypic representation G and a phenotypic representation $P \simeq D$
3. Choose a generator function γ
4. Choose aesthetic learning function λ_a
5. Choose a novelty measure ν_g
6. Choose genotype evaluation learning function ϕ_g
7. Choose phenotype evaluation learning function ϕ_p
8. $t = 0$
9. While not done
 - $t = t + 1$
 - Collect data and build knowledge base K^t
 - $A^t \leftarrow \lambda_a(K^t)$
 - $v_g^t \leftarrow \phi_g(A^t)$
 - construct a genotypic evaluator ϵ_g^t from ν_g and v_g^t
 - $\epsilon_p^t \leftarrow \phi_p(A^t)$
 - $C^t \leftarrow \lambda_c(K^t)$
 - While $\epsilon_p^t(p) < \theta_p$
 - While $\epsilon_g^t(g) < \theta_g$
 - $g \leftarrow \gamma(C^t, \rho)$
 - $p \leftarrow \tau(g)$
 - return(p)

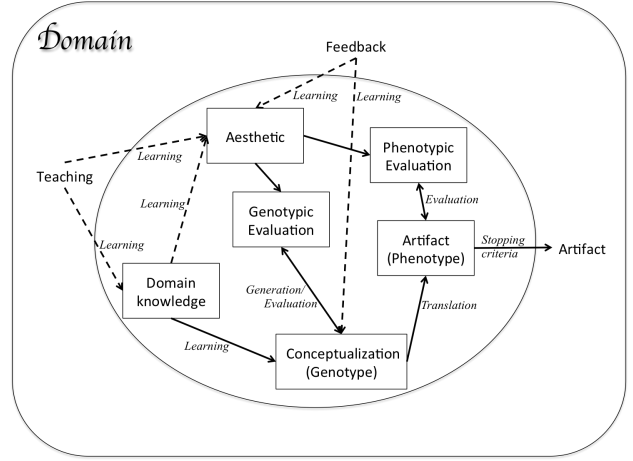


Figure 3: For increased autonomy, and therefore, presumably increased (potential) creativity, a CC system should not only be embedded in a domain but should also be capable of interacting with its embedding domain. This interaction will most often take form of teaching and feedback that allows the system both to dynamically update its background knowledge base and aesthetic as new domain information becomes available and to adapt its aesthetic, conceptualization and generation mechanisms based on feedback it receives in response to the artefacts it creates (and releases externally).

(The formal view of this does not change in an interesting way, so it will not be repeated here with only minor changes.)

Note the shift towards greater autonomy in the fact that the *a priori* design decisions (made initially, before the operational loop) are being made for more abstract entities (i.e., designing functions for learning components rather than designing the components themselves). Also, note that some of this shift towards greater autonomy is not strictly necessary in the sense that design decisions that make some system component time-independent are certainly still possible. On the other hand, it is also possible to consider designing additional abstract learning functions that allow additional components to become time dependent (e.g., γ , λ_a , λ_c , ν_g , τ , θ_g , θ_p).

Intentionality

Though novelty and value have been incorporated into the construction or learning of the evaluation functions, system intentionality has not yet been explicitly addressed. In what way can a system built using the proposed approach be claimed to be intentional?

The answer is first that, in a limited sense, the system has a goal to produce novel, valuable artefacts in the domain D . This may not be entirely satisfactory to critics who may argue (correctly) that the goal is imposed on the system by an external agent (the designer); however, the definition of intentionality offered in the introduction does not require that the system invent its own goals (see the discussion on Turtles

below).

Another way the system may be said to exhibit intentionality is if its product (the artefacts it produces) is correlated with its process (how it produces them). This is certainly the case, by design, especially with respect to, for example, the aesthetic-based evaluation mechanisms—the system has intention to some extent because to some extent it “understands” what it is creating.

Yet another possible indicator for intention, and one not explicitly included in the CC system building process proposed here, is an ability of the system to explain its process and/or product—can the system justify in some way why it made the decisions it made and why the result is what it is? This is one example of the broad notion of *framing*, and though not required as part of the blueprint provided here, it is often desirable and sometimes not too difficult to incorporate some basic framing ability in CC systems (e.g., showing source material for musical inspiration, explaining perception of images, showing the connection between setup and punchline, etc.), and when this is done, it provides further support for a claim of system intentionality.

(External) Evaluation

Once the system is running and producing artefacts, at some point the question must be asked whether it is doing so satisfactorily. Of course, from the system’s perspective, this question has already been answered in the affirmative—by design, any artefact produced has already been vetted (in two different ways) and found to meet aesthetic and uniqueness standards. However, this is not sufficient in the sense that creation and contribution to a domain are inherently social processes—a creator can not be the sole arbiter when it comes to the question of its creativity. Therefore, it becomes important to subject the system to external scrutiny. While how best to do this is still an open question, there have been multiple proposals for evaluation mechanisms for CC systems. Collectively, these can examine both system product and process and include Ritchie’s suggestions for formally stated empirical criteria focusing on the relative value and novelty of system output (2007); the FACE framework for qualifying different kinds of creative acts performed by a system (Colton, Charnley, and Pease 2011); the SPECS methodology which requires evaluating the system against standards that are drawn from a system specification-based characterization of creativity (Jordanous 2012); and Ventura’s proposed spectrum of abstract prototype systems that can be used as landmarks by which specific CC systems can be evaluated for their relative creative ability (2016).

A Note on Appropriation

Pablo Picasso is often credited with having said, “good artists borrow; great artists steal”. In the context of building a CC system, it is perhaps stating the obvious to suggest that one should avoid re-inventing the wheel when possible. There are a great number of existing tools, web services, and APIs that can be incorporated into the architecture of a CC system as solutions for many of the components discussed here. In many cases, these tools are exactly what one

is wanting and may be found with some little effort (e.g., the many extant NLP and computer vision tools freely available on the web); in other cases, useful resources exist, but may need to be discovered more serendipitously or cleverly re-purposed (e.g., using an online list of “Top Tens” as a source of pop culture references). The take away here should be that it is often worth spending significant time searching for existing resources instead of trying to build them from scratch—it is very often the case that someone has already done the work and done it well. Commonly, the main (engineering) contribution of a CC system is not in the implementation of components but in the system building and architectural work.

A Note on Turtles

It should be noted here that the proposed process for CC system building does not incorporate state-of-the-art thinking and address all the latest questions that are being asked in the field, and that is not its intention. However, it does perhaps come close to encapsulating the state-of-the-art with respect to actual working CC systems currently being built.

There is certainly a great deal more that can be (and is being) said about the topic, and, in particular, it should be mentioned that the field of CC is by nature begging for treatment at the meta-level. That is, it is fair game to consider the domain D of knowledge representations or conceptual models or aesthetics or learning functions or evaluation functions or goals, and to then attempt to build a CC system embedded in *that* domain—a system for inventing new and useful representations or conceptual models or aesthetics or learning/evaluation functions or goals. One could then consider building a hierarchical system that incorporates meta-level creativity to improve base-level creativity, increasing the overall system autonomy and further distancing the (eventual) creative system from the original designer.

Of course, this begs the question of meta-meta-level considerations, but it is not clear that such thinking is useful, either for CC systems, nor perhaps even with respect to human cognition, and in any case, it is beyond the scope of the current discussion.

Conclusion

The main contribution of this paper is a general recipe for the construction of an autonomous, computationally creative agent, with the intention being that anyone can follow the proposed process to build a CC agent for an arbitrary domain. Of course, domain-specific challenges are not addressed here and remain as opportunities for the system builders to act creatively themselves. A secondary contribution of the paper is the initiation of a conversation about whether or not the main contribution is realizable—is it possible to describe a useful, general-purpose approach to CC system building?

References

Bickerman, G.; Bosley, S.; Swire, P.; and Keller, R. M. 2010. Learning to create jazz melodies using deep belief nets. In Ventura, D.; Pease, A.; Pérez y Pérez, R.; Ritchie,

- G.; and Veale, T., eds., *Proceedings of the International Conference on Computational Creativity*, 228–237.
- Binsted, K., and Ritchie, G. 1994. A symbolic description of punning riddles and its computer implementation. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, 633–638.
- Colton, S.; Bundy, A.; and Walsh, T. 1999. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 786–791.
- Colton, S.; Charnley, J.; and Pease, A. 2011. Computational creativity theory: The FACE and IDEA descriptive models. In *Proceedings of the 2nd International Conference on Computational Creativity*, 90–95.
- Colton, S. 2012. The Painting Fool: Stories from building an automated painter. In McCormack, J., and D’Inverno, M., eds., *Computers and Creativity*. Berlin, Germany: Springer-Verlag. 3–38.
- Cook, M.; Colton, S.; and Gow, J. 2016. The ANGELINA videogame design system, part I. *IEEE Transactions on Computational Intelligence and AI in Games* to appear.
- de Silva Garza, A. G., and Maher, M. 2000. A process model for evolutionary design case adaptation. In Gero, J., ed., *Proceedings of the Sixth International Conference on Artificial Intelligence in Design*, 393412. Kluwer Academic Publishers.
- Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4(3):246–279.
- Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2012. Adapting models of visual aesthetics for personalized content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 4(3):213–228.
- Love, N.; Hinrichs, T.; Haley, D.; Schkufza, E.; and Genssereth, M. 2006. General game playing: Game description language specification. Lg-2006-01, Stanford.
- Morris, R.; Burton, S.; Bodily, P.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *Proceedings of the 3rd International Conference on Computational Creativity*, 119–125.
- Norton, D.; Heath, D.; and Ventura, D. 2013. Finding creativity in an artificial artist. *Journal of Creative Behavior* 47(2):106–124.
- Oliveira, H. G. 2012. PoeTryMe: a versatile platform for poetry generation. In *Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence*.
- Özbal, G.; Pighin, D.; and Strapparava, C. 2013. BRAIN-SUP: Brainstorming support for creative sentence generation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1446–1455.
- Pachet, F., and Roy, P. 2014. Non-conformant harmonization: the real book in the style of Take 6. In *Proceedings of the 5th International Conference on Computational Creativity*, 100–107.
- Pérez y Pérez, R., and Sharples, M. 2004. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems* 17(1):15–29.
- Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):217–268.
- Ritchie, G. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17:67–99.
- Smith, M. R.; Hintze, R. S.; and Ventura, D. 2014. Nehovah: A neologism creator nomen ipsum. In *Proceedings of the 5th International Conference on Computational Creativity*, 173–181.
- Stock, O., and Strapparava, C. 2003. HAHAAcronym: Humorous agents for humorous acronyms. *Humor - International Journal of Humor Research* 16(3):297–314.
- Strapparava, C.; Valitutti, A.; and Stock, O. 2007. Automating two creative functions for advertising. In *Proceedings of 4th International Joint Workshop on Computational Creativity*, 99–105.
- Toivanen, J. M.; Toivonen, H.; Valitutti, A.; and Gross, O. 2012. Corpus-based generation of content and form in poetry. In *Proceedings of the 3rd International Conference on Computational Creativity*, 175–179.
- Varshney, L.; Pinel, F.; Varshney, K.; Schorgendorfer, A.; and Chee, Y.-M. 2013. Cognition as a part of computational creativity. In *Proceedings of the 12th IEEE International Conference on Cognitive Informatics and Cognitive Computing*, 36–43.
- Veale, T., and Hao, Y. 2007. Comprehending and generating apt metaphors: A web-driven, case-based approach to figurative language. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 1471–1476.
- Veale, T. 2013. Less rhyme, more reason: Knowledge-based poetry generation with feeling, insight and wit. In Maher, M. L.; Veale, T.; Saunders, R.; and Bown, O., eds., *Proceedings of the Fourth International Conference on Computational Creativity*, 152–159.
- Ventura, D. 2016. Mere generation: Essential barometer or dated concept? In Pachet, F.; Cardoso, A.; Corruble, V.; and Ghedini, F., eds., *Proceedings of the Seventh International Conference on Computational Creativity*, 17–24.

Generating Animations by Sketching in Conceptual Space

Tom White*, Ian Loh*

School of Design
Victoria University of Wellington
Wellington, New Zealand

{tom.white@vuw.ac.nz, lohjun@myvuw.ac.nz}

* equal contribution

Abstract

We introduce a new sketch based interface for generating animations. Unlike traditional digital tools, ours is parameterized entirely by a neural network with no pre-programmed rules or knowledge representations. The capability of our sketching tool to support visual exploration and communication is demonstrated within the context of facial images, though our framework is domain independent. Our recorded sketches serve not only as a means for generating a specific animation, but also as a standalone visual encapsulation of an animation's semantic operation which can be reused and refined.

Introduction

Sketching is the process of quickly exploring an idea through rough designs that focus on key details. In animation, drawings such as thumbnail sketches and pencil tests are used to study the flow of movement. These drawings are often gestural in nature, with loose lines that capture qualities of the animation's structure and its movement. The affordance of speed while still communicating essential qualities makes this process of gestural sketching an ideal tool for supporting digital animation workflows, where high fidelity, production-orientated interfaces may impose obstacles to ideation (Rettig 1994).

Many proposed systems utilise gesture based input in animation sketching by mapping the user's actions to character movement. Building on this, we have developed TopoSketch, a tool for prototyping the animation of faces by sketching gestures using a tablet pen or mouse (Figure 1). A notable difference is that we have displayed gestures as a drawing, intended as a form of encoding that visually communicates qualities of the movement. Gestures generate a path through a vector space of faces generated by a neural network - called a conceptual space - that abstracts the complex task of posing faces into a simple animation controller. TopoSketch animations are generated by alternatively exploring and charting a path through a pre-defined topographic conceptual space.

There are aesthetic and functional motivations to exploring gestural input for digital animation sketching. Abstract gestural lines have been used by animators as a mental model for studying and planning movement, such as "whips" or "lines of action" (Williams 2001; Blair 1994).

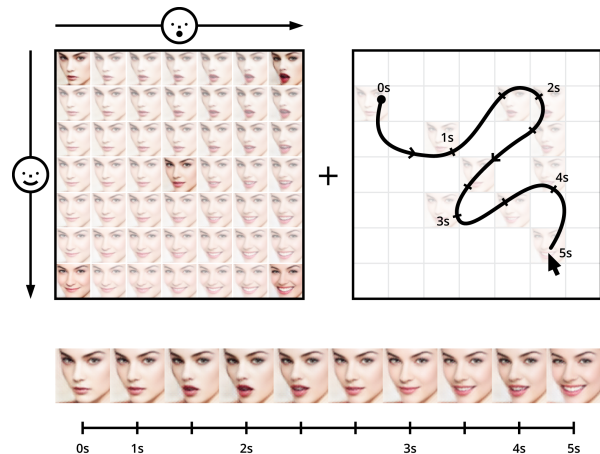


Figure 1: TopoSketch animation process

The director of the hand drawn movie *Persepolis* (Paronnaud and Satrapi 2007) describes that the 'vibrations of the hand make the drawings come to life' (Education 2007), using these natural variations as a storytelling device. The ability digitally impart these gestures may lend a reflexive element to animation, facilitating a more expressive intuition of the work (Power 2009).

Our system is unique in that it is parameterized not by a rule based system, but instead by the geometric representation layer of the conceptual space. The conceptual space is built from a neural network trained to reconstruct images. This parameterization is reconfigurable and the artifacts produced support and extend sketching as a medium of visual exploration and communication. Additionally those sketches become reusable components of the overall workflow. The sketch's appearance serves as a visual mnemonic to suggest the types of animations that will result when used as an operation in the conceptual space, and the operation the sketch represents can be also reused to achieve a similar animation on different input images.

Background

Sketch and Gesture Animation Interfaces

TopoSketch builds upon other systems that utilise sketch and gesture based input for animation. These applications fall broadly into two categories: the posing of objects or characters, and the generation of movement of those objects.

Conventional animation systems are production orientated, enabling control over many small aspects of an animation. However, a high degree of control is overwhelming and is not ideal for earlier stages of ideation (Rettig 1994). Sketch and gesture based input offer a unique approach to this problem, as its looser precision and intuitive mode of input are suitable qualities for informal tools, letting users focus on the larger picture (Igarashi 2003; Zeleznik, Herndon, and Hughes 1996).

It should be noted the word "sketch" is used in this paper to generally refer to any form of mark making done quickly to explore an idea. This can include, but should not be confused with a "design sketch", which typically focuses on structural representation, such as the shape of a car. Our definition also includes our gesture visualisation as a type of sketch, along with other notations with less literal representations.

Numerous sketch based interfaces have been proposed to make the task of posing characters easier and more accessible. Typically this involves mapping a series of drawn guides as a set of deformations applied to an object. Some enable granular adjustments through familiar notations such as stick figure drawings (Davis et al. 2003; Matthews and Vogts 2011), or by direct drawing on the model itself (Chang and Jenkins 2006). However, the design intention of our system more closely resembles the level of detail afforded by Guay, Cani, and Ronfard, where a single stroke or "line of action" is used to control the pose of the entire character (Guay, Cani, and Ronfard 2013). The authors argue that this abstraction is less time consuming, and allows users to focus on the overall expressiveness of the animation, better reflecting cognitive workloads involved in early stages of animation. TopoSketch also shares similarities to Sketch Express, where a 2D control interface consisting of a window containing separate drawing regions control different parts of a face (Miranda et al. 2011). While Sketch Express' standardisation lets the same poses to be reused on different faces, our grid based sketches retargets animation timings to different sets of facial expressions.

Similar to posing, sketch based interfaces for generating movement involve recording a path traced by the user - such as a mouse or tablet - and mapping it to an object's transformation. One approach is based on direct manipulation - such as dragging an object across the screen - while the system records these changes (Moscovich and Hughes 2001; Davis, Colwell, and Landay 2008).

Another technique is performance based timing, where the movement of a user drawn path is used to progress through a set of predefined keyframes (Terra and Metoyer 2004) and (Walther-Franks et al. 2012). This allows users to act out their animations, while retaining the precision of keyframes.

Spatial keyframes incorporate aspects of both techniques - users place keyframes with different poses within a scene. Animations are created by moving a cursor in between the spatial keyframes, blending the different poses together based on their distances (Igarashi, Moscovich, and Hughes 2005). The effect of mapping user's movements to complex poses give a "puppeteering" feel to the system. The authors note that resulting animations are able to make apparent the user's natural sense of timing, contributing to a unique aesthetic. The feeling of creating an animation in TopoSketch is similar in spirit to spatial keyframes, as our animation window can be thought of as having four "keyframes", one in each corner. However, we do not allow the creation of new "keyframes" for more customised controls.

In all approaches, human factors such fatigue, physical limits and acting ability all affect the complexity, quality and length of the animation. For example, it is unreasonable to expect a person to act out a five minute animation in one go. We address this by allowing users to scrub through the timeline to overwrite an area of animation, or continue where they left off.

Another style of sketch based animation generation is through the use of notations. Users draw symbols on top of the scene, which are then parsed into a series contextual animations for a character based on the symbol's position and shape (Thorne, Burke, and van de Panne 2004; Jang et al. 2014; Kazi et al. 2014). While the notations are limited to the number of available symbols, they are often iconic in nature (such as arrows and loops), providing a meaningful visual record of the animation. Although less descriptive in comparison, we argue that our visualised gestures are still able to describe high level aspects of an animation.

In the domain of neural network based tools, sketch input has been used to facilitate searching and exploration of latent spaces. In image manipulation tools by Zhu et al. and Brock et al., instead of directly changing pixels, users are given a "contextual paintbrush" to draw guiding marks on the image. This rough drawing is used to indirectly navigate a space of generated images within a smaller manifold of coherent results. These assisted interfaces act as "safety wheels" that allow novice users to make unsupervised changes while maintaining plausible outputs (Zhu et al. 2016; Brock et al. 2016).

Conceptual Spaces

Generative models are a popular approach to unsupervised machine learning. Generative neural network models are trained to produce data samples that resemble the training set (Karpathy et al. 2016). Because the number of model parameters is significantly smaller than the training data, the models are forced to discover efficient data representations. These models are sampled from a set of latent variables in a high dimensional space, called a latent space. Latent space can be sampled to generate observable data values. Learned latent representations often also allow semantic operations with vector space arithmetic (Figure 2), a phenomenon discovered previously in the latent space of language models (Mikolov et al. 2013).

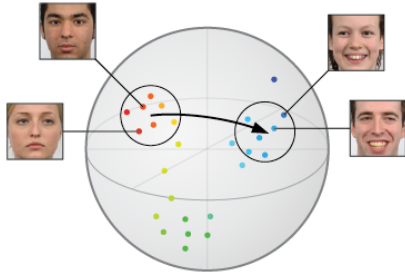


Figure 2: Schematic of the latent space of a generative model. In the general case, a generative model includes an encoder to map from the feature space (here images of faces) into a high dimensional latent space. Vector space arithmetic can be used in the latent space to perform semantic operations. The model also includes a decoder to map from the latent space back into the feature space, where the semantic operations can be observed. If the latent space transformation is the identity function we refer to the encoding and decoding as a reconstruction of the input through the model.

Generative models are often applied to datasets of images. Two popular generative models for image data are the Variational Autoencoder (Kingma and Welling 2013) (VAE) and the Generative Adversarial Network (Goodfellow et al. 2014) (GAN). VAEs use the framework of probabilistic graphical models with an objective of maximizing a lower bound on the likelihood of the data. GANs instead formalize the training process as a competition between a generative network and a separate discriminative network. Though these two frameworks are very different, both construct high dimensional latent spaces that can be sampled to generate images resembling training set data. Moreover, these latent spaces are generally highly structured and can enable complex operations on the generated images by simple vector space arithmetic in the latent space (Larsen, Sønderby, and Winther 2015).

In the latent space of generative models, many high level attributes can be represented as a vector (Figure 3). Using techniques from (White 2016), multiple attributes can be decoupled further to create a visualization of possible states across multiple semantic vectors (Figure 4). For example, when trained on a dataset of portraits, latent vectors can be computed for "smiling" and "mouth open" which then applied to new face images.



Figure 3: Traversals along the smile vector using a GAN model from (Dumoulin et al. 2016)

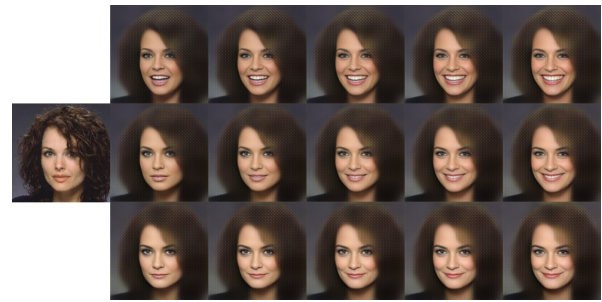


Figure 4: Decoupling attribute vectors for smiling (x-axis) and mouth open (y-axis) allows for more flexible latent space transformations. Input shown at left with reconstruction adjacent. Using a VAE model from (Lamb, Dumoulin, and Courville 2016)

Prior to the discovery of neural network latent spaces supporting semantic operations, cognitive science had hypothesized the existence of knowledge representations that were primarily geometric instead of symbolic. One primary proponent was Gärdenfors who proposed a framework of "Conceptual Spaces" as structured multi-dimensional feature spaces to support modeling information processes such as concept learning and prototype theory (Gärdenfors 2011). Notably, conceptual spaces were proposed as a model of how people structure concepts, independent of any proposed computational implementation of how they might come about.

We adapt the terminology and claim that latent spaces of generative neural networks function as conceptual spaces which can be used as a non-symbolic knowledge representation layers in other tools. Conceptual spaces are a useful medium for building human-centered tools as compared with the "black-box" neural network systems which lack useful substructures or the more brittle symbolic approaches of rule based systems.

With this framework, we examine the ability of a geometric representation layer built from the latent space of a generative neural network model to support a new type of sketching interface tool. In our initial iteration, we explore the conceptual space of human faces, but the tool itself is domain independent and could be used on other similar domains. In exploring the domain of human faces, our tool constructs subspaces of the larger conceptual space of human portraits as a parameter space of a sketch driven animation tool.

TopoSketch

TopoSketch is a sketch based facial animation tool that uses neural networks to navigate a plausible animation manifold. Posing and animating a believable face is a complex process, due to the interrelation of different facial features (eg: the eyes narrowing during a smile). While other systems allow posing of individual facial features, TopoSketch uses utilities a higher level control grid based on expressions such as 'smiling', or 'opening mouth'. These expressions represent changes of many separate features simultaneously. Drawn

gestures are then used to control the interpolation and timing between these different expressions.

The current TopoSketch prototype is based on a VAE model described in (Lamb, Dumoulin, and Courville 2016). Our model is initially trained in an unsupervised fashion on images from the CelebA training set (Liu et al. 2015) resized to 256x256. After training, concept vectors are built using attributes from both the CelebA and the Radboud Faces Database (Langner et al. 2010). We have extended the techniques of constructing concept vectors as described in (White 2016) by also using vectors orthogonal to an SVM hyperplane in latent space, which we have found gives superior results when given sufficient training data.

Extended information covering our current TopoSketch implementation including videos is available online.¹

Workspace

The main TopoSketch workspace is organized into two windows side by side: the animation window and preview window (Figure 5). When an image of a face is loaded into the tool, it is processed by TopoSketch and an animation grid of generated faces is displayed in the animation window. The x and y axis of this grid each represent the results of different operations applied to the input face, increasing in effect along the axis. For example, the face’s mouth widens and smile gets larger along x and y axis respectively. Animations are recorded by drawing gestures within the animation window. Animation playback is controlled using a timeline located beneath both windows. Buttons above the windows provide additional functions for exporting, loading and clearing animations or grids.

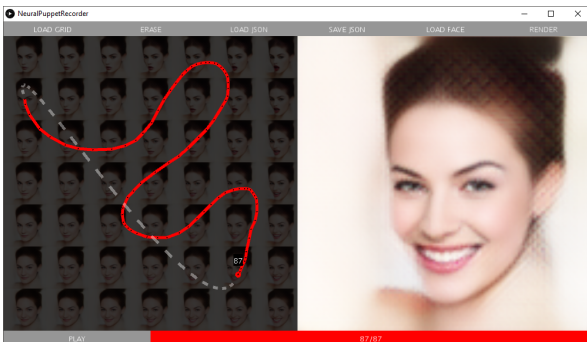


Figure 5: TopoSketch in use

Creating Animations

When the cursor is placed within the animation window, the face closest to the cursor on the grid is shown in the preview window. The user can move the cursor over the animation grid to create gestures, ”scrubbing” through the animation grid to create transitions between the faces. Moving along a single axis will only affect the corresponding operation (eg. only smiling) while moving in both axes changes the face with both operations. Once a suitable gesture has

¹<https://vUSD.github.io/toposketch/>

been found, the cursor’s movement can be recorded by click-dragging within the animation window. The movement is recorded in real time, with the cursor position recorded 25 times a second. This recording is displayed as a line over the animation grid as the user draws, allowing them to see how the animation has progressed. Releasing the cursor stops the recording. To create a smooth loop, the start and end points of the recording are automatically joined with a Bezier curve.

Editing Animations

TopoSketch currently supports basic editing capabilities such as erasing the recording, jumping to any particular time, and continuing the recording from that point. Animations are stored in a modular path file and the animation grid image is an interchangeable element. Paths can be exported for use in another animation or further refinement in another program. Path files can also be rendered offline by fully interpolating and sampling in the model’s latent space for more temporal resolution. Different animation grids can also be loaded into TopoSketch to reuse the effect of an existing animation with either new faces or conceptual spaces.

Discussion

TopoSketch proposes a method of creating facial animations through a very high level, sketch based interface. Neural network generated conceptual spaces provide an underlying ”intuition” that allows simple gestural strokes to be translated into feasible looking transitions between different face expressions. We aspire that the affordances of this style of tool could be useful in a number of practical and aesthetic exploratory applications. The combination of quick low precision gestures, simple representation, and low investment of face posing creates an environment that supports weak filters for quality, encouraging experimentation (Kim, Bagla, and Bernstein 2015).

Many aspects of our system are modular, as both gestures and faces are interchangeable. While naïve gesture and face combinations may not yield practical results, a similar system by (Igarashi, Moscovich, and Hughes 2005) suggests transferring gestures would be useful within similar classes of motion. Examples of these classes can be seen in guides used to plan animation timing, such as ”whips” and ”waves”, that are general enough to be applied to a variety of use cases. For example: both batting eyelashes and an expressive laugh both use an underlying ”whip” gesture (Figure 6). The same expression can also vary based on factors such as age and or stress. By changing our conceptual space, we are able to compare the nuances present in these different situations (eg: stressed smile versus a relaxed one). The effects can also be made more extreme, for a caricature-like effect. This can be used as an underlying guide in animation workflows, and for exploring more diverse expressions. While our faces are photorealistic, different stylistic results may be obtained by employing a different model, such as one trained on line drawings. We currently do not provide a way in the tool for users to customise the parameters of the animation grid. However, we envision a mature version of this tool could have a library of expressions that users can

browse from, or custom expression creation using a webcam.



Figure 6: A sketch demonstrating a "whip" action in motion (left) being applied to the head of a laughing character (right) and batting eyelashes (bottom). (Williams 2001)

Our animation workflow is much more reflexive compared to conventional systems, where animators go back and forth between setting keyframes, and playing back the animation changes. In TopoSketch, animation is created in real-time and viewed in tandem, allowing many different gestures to be explored quickly and practiced, before committing to a final recording. Being able to "act out" or "puppeteer" faces using gestures allows users to make reactive adjustments as they are sketching, leading to some stylistic affordances that are not easy to do in conventional tools. For example, start-stop movements that are based on the previous position, or repetitive actions that vary naturally over time. This animation style can potentially be compared to motion capture, or techniques such as straight-ahead animation, which encourage more spontaneous movements (Lasseter 1987).

Displaying recorded gestures as a drawing may have potential applications as a communication aid. While ours does not specifically describe the contents of an animation such as (Thorne, Burke, and van de Panne 2004) or (Kazi et al. 2014), our gestures can still provide some context on the type of the movement. For example, a jagged drawing indicates sudden changes in expression while smoother gestures indicate gentler transitions. Animators already employ similar abstract gestures as guides to study motion. Exposing the visual qualities of more types of animations may lead to serendipitous ideas by way of gestalt effect, or seeing images within the drawings (Owen 2012).

Neuroscience research suggests that being able to see artifacts such as brushstrokes can evoke empathetic responses in viewers (Freedberg and Gallese 2007). While the "mark", or underlying structure is quite visible with traditional animation processes (such as guidelines), there is a lack of such in computer animation. Our displayed gestures can be seen to facilitate such a mark, by exposing the construction of movement. Power suggests these indexical artifacts may enable animators to "feel the movement behind the mark" (Power 2009), opening up a different way to perceive anima-

tions. Applications of this can include comparing work from different artists, or as a classification technique for large animation sets. In practice, we have adopted drawn gestures into a notation for planning and describing animations on paper, a formalized version of which is employed in the figures.

Future Work

We plan to expand the concept of the animation grid to accommodate more user customization and potential operation combinations. As an alternative to the grid format, we are exploring using geometric shapes to define how operations are distributed. Adapting TopoSketch to a more freeform interface used in our previous research (Loh 2017) could allow for customized layouts supporting a wider range of animation possibilities.

TopoSketch is able to create a wide range of operations that go beyond facial expressions (eg. getting older or putting on sunglasses). In addition to encouraging users to provide their own images to be used as the subject of an animation, we are also exploring allowing one-shot training of custom facial operations by accepting reference image pairs to define new concept vectors.

References

- Blair, P. 1994. *Cartoon Animation*. Tustin, CA: Walter Foster Publishing.
- Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2016. Neural photo editing with introspective adversarial networks. *CoRR* abs/1609.07093.
- Chang, E., and Jenkins, O. C. 2006. Sketching articulation and pose for facial animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, 271–280. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.
- Davis, J.; Agrawala, M.; Chuang, E.; Popović, Z.; and Salesin, D. 2003. A sketching interface for articulated figure animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, 320–328. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.
- Davis, R. C.; Colwell, B.; and Landay, J. A. 2008. K-sketch: A 'kinetic' sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, 413–422. New York, NY, USA: ACM.
- Dumoulin, V.; Belghazi, I.; Poole, B.; Mastropietro, O.; Lamb, A.; Arjovsky, M.; and Courville, A. 2016. Adversarially Learned Inference. *ArXiv e-prints*.
- Education, F. 2007. Persepolis Production Process.
- Freedberg, D., and Gallese, V. 2007. Motion, emotion and empathy in esthetic experience. *Trends in Cognitive Sciences* 11(5):197–203.
- Gärdenfors, P. 2011. Semantics based on conceptual spaces. In *Logic and Its Applications - 4th Indian Conference, ICLA 2011, Delhi, India, January 5-11, 2011. Proceedings*, 1–11.

- Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2672–2680.
- Guay, M.; Cani, M.-P.; and Ronfard, R. 2013. The line of action: An intuitive interface for expressive character posing. *ACM Trans. Graph.* 32(6):205:1–205:8.
- Igarashi, T.; Moscovich, T.; and Hughes, J. F. 2005. Spatial keyframing for performance-driven animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '05*, 107–115. New York, NY, USA: ACM.
- Igarashi, T. 2003. Freeform user interfaces for graphical computing. In *Proceedings of the 3rd International Conference on Smart Graphics, SG'03*, 39–48. Berlin, Heidelberg: Springer-Verlag.
- Jang, H.; Jeon, J.; Sohn, E.; Lim, S.-B.; and Choy, Y.-C. 2014. A sketch-based interface to modify and reproduce motion sequences. *Multimedia Tools Appl.* 72(1):591–612.
- Karpathy, A.; Abbeel, P.; Brockman, G.; Chen, P.; Cheung, V.; Duan, R.; Goodfellow, I.; Kingma, D.; Ho, J.; Houthoofd, R.; Salimans, T.; Schulman, J.; Sutskever, I.; and Zaremba, W. 2016. Generative models. Technical Report 1, OpenAI.
- Kazi, R. H.; Chevalier, F.; Grossman, T.; Zhao, S.; and Fitzmaurice, G. 2014. Draco: Sketching animated drawings with kinetic textures. In *ACM SIGGRAPH 2014 Studio, SIGGRAPH '14*, 34:1–34:1. New York, NY, USA: ACM.
- Kim, J.; Bagla, A.; and Bernstein, M. S. 2015. Designing creativity support tools for failure. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition, C&C'15*, 157–160. New York, NY, USA: ACM.
- Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114.
- Lamb, A.; Dumoulin, V.; and Courville, A. 2016. Discriminative Regularization for Generative Models. *ArXiv e-prints*.
- Langner, O.; Dotsch, R.; Bijlstra, G.; Wigboldus, D.; Hawk, S.; and van Knippenberg, A. 2010. Presentation and validation of the radboud faces database. In *Cognition and Emotion*.
- Larsen, A. B. L.; Sønderby, S. K.; and Winther, O. 2015. Autoencoding beyond pixels using a learned similarity metric. *CoRR* abs/1512.09300.
- Lasseter, J. 1987. Principles of traditional animation applied to 3d computer animation. *SIGGRAPH Comput. Graph.* 21(4):35–44.
- Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
- Loh, I. 2017. Anisketch: Alternative aesthetics for computer animation tools. Master's thesis, School of Design, Victoria University of Wellington, Wellington, NZ.
- Matthews, T., and Vogts, D. 2011. A sketch-based articulated figure animation tool. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment, SAICSIT '11*, 151–160. New York, NY, USA: ACM.
- Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.
- Miranda, J. C.; Alvarez, X.; Orvalho, J. a.; Gutierrez, D.; Sousa, A. A.; and Orvalho, V. 2011. Sketch express: Facial expressions made easy. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, SBIM '11*, 87–94. New York, NY, USA: ACM.
- Moscovich, T., and Hughes, J. F. 2001. Animation sketching: An approach to accessible animation. Master's thesis, Department of Computer Science, Brown University, Providence, RI.
- Owen, A. 2012. Neuroanimatics: Hyperrealism and digital animations accidental mark. *Animation Practice, Process and Production* 1(2):315–345.
- Paronnaud, V., and Satrapi, M. 2007. Persepolis.
- Power, P. 2009. Animated expressions: Expressive style in 3d computer graphic narrative animation. *Animation* 4(2):107–129.
- Rettig, M. 1994. Prototyping for tiny fingers. *Commun. ACM* 37(4):21–27.
- Terra, S. C. L., and Metoyer, R. A. 2004. Performance timing for keyframe animation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04*, 253–258. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.
- Thorne, M.; Burke, D.; and van de Panne, M. 2004. Motion doodles: An interface for sketching character motion. *ACM Trans. Graph.* 23(3):424–431.
- Walther-Franks, B.; Herrlich, M.; Karrer, T.; Wittenhagen, M.; Schröder-Kroll, R.; Malaka, R.; and Borchers, J. 2012. Dragimation: Direct manipulation keyframe timing for performance-based animation. In *Proceedings of Graphics Interface 2012, GI '12*, 101–108. Toronto, Ont., Canada, Canada: Canadian Information Processing Society.
- White, T. 2016. Sampling generative networks. *CoRR* abs/1609.04468.
- Williams, R. 2001. *The Animator's Survival Kit*. London: Faber.
- Zelevnik, R. C.; Herndon, K. P.; and Hughes, J. F. 1996. Sketch: An interface for sketching 3d scenes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96*, 163–170. New York, NY, USA: ACM.
- Zhu, J.-Y.; Krähenbühl, P.; Shechtman, E.; and Efros, A. A. 2016. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*.

Appendix: Example Animations

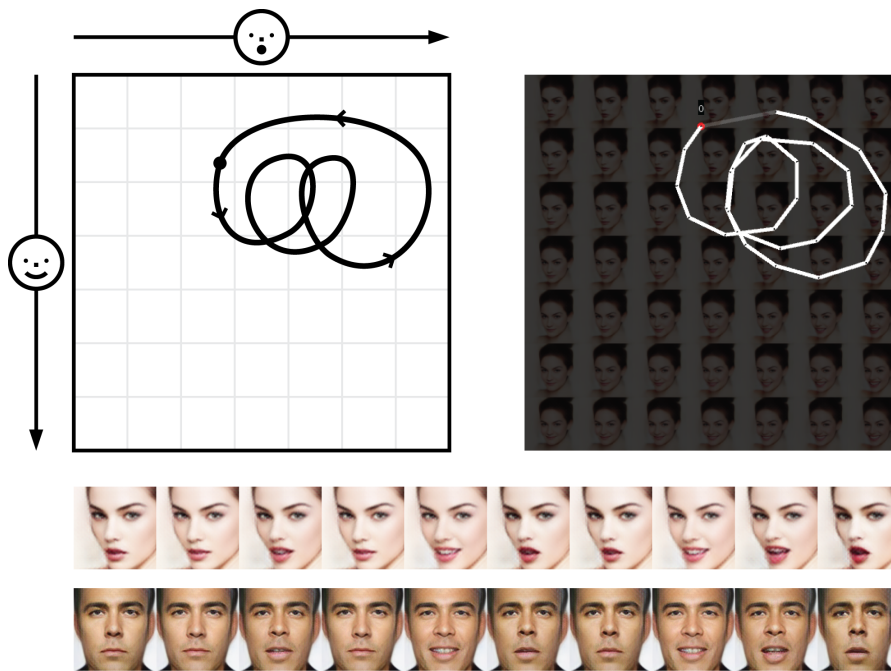


Figure 7: Chewing Animation. The loops in the sketched sequence indicate repeated motions in the animation. The sketch gesture can also be transferred to a second input image without modifying the path.

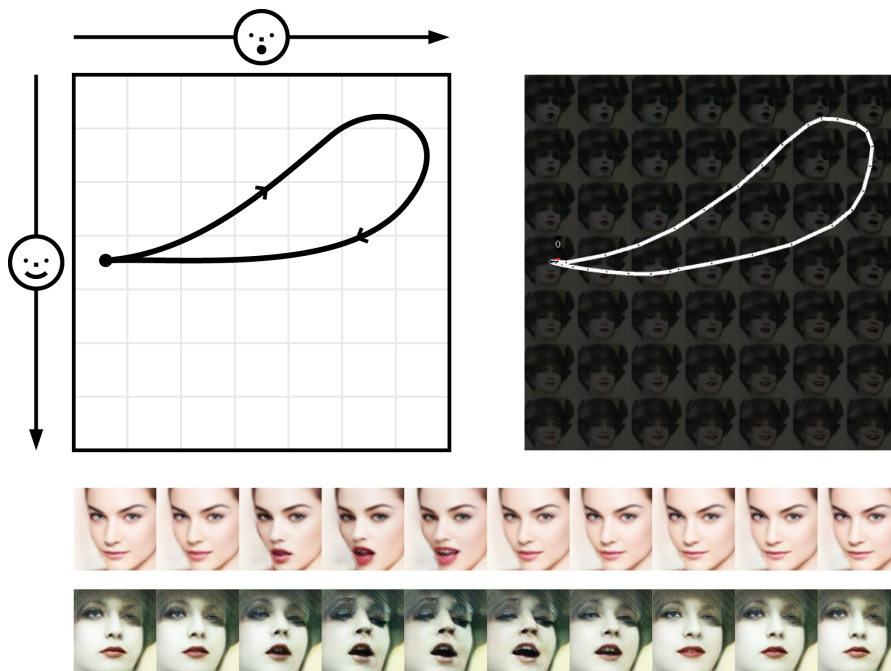


Figure 8: Kissing Animation shows how a sketched sequence is reused on a second input image. The captured sketch is a reusable component that can be applied to other inputs.

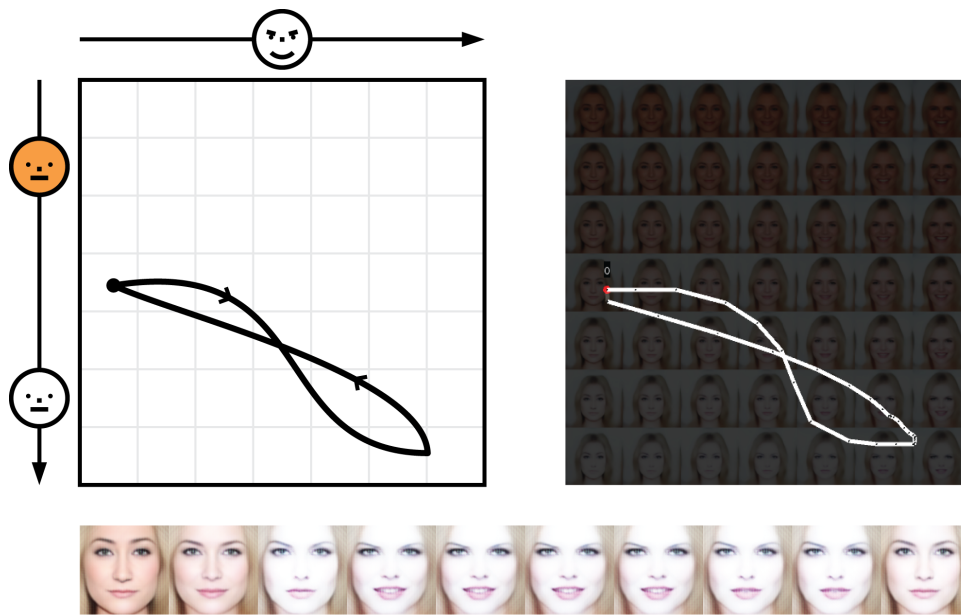


Figure 9: Evil Grin Animation. Depending on the intent, the attributes represented in the conceptual subspace can be changed. In this example, a subspace is created which combines disgust, smiling, and skin tone.

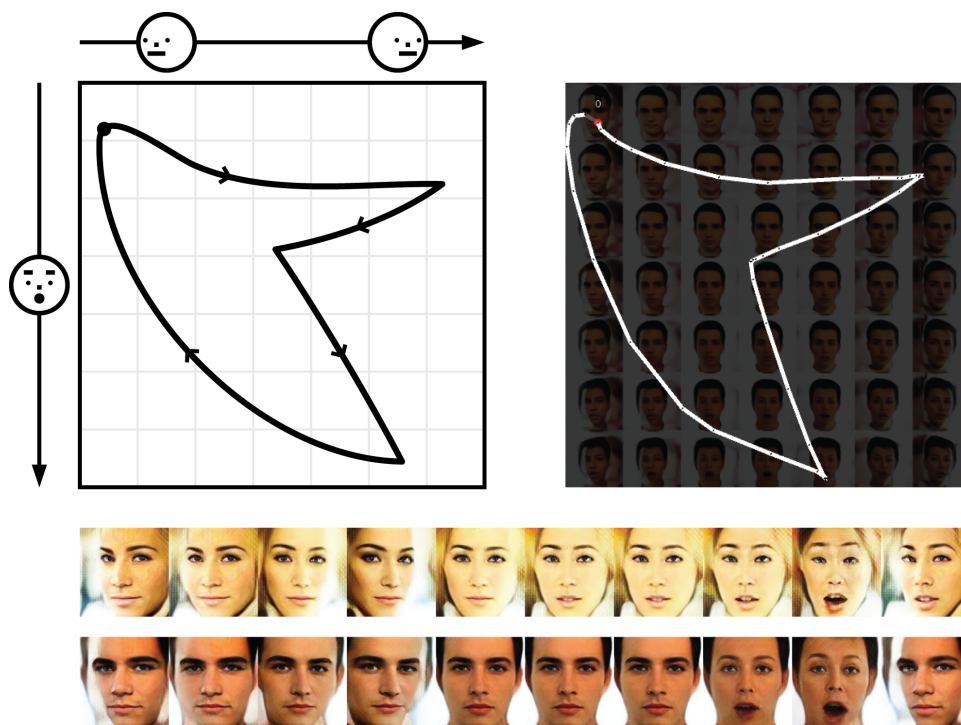


Figure 10: Parameters for animations can combine facial expressions with other changes such as orientation and lighting. Here a "double take" animation is constructed from face rotation and an expression of surprise.

