StoReys: A neurosymbolic approach to human-AI co-creation of novel action-oriented narratives in known story worlds

Alaeddine Mellouli, Rahul Chaudhari
Chair of Media Technology
Technical University of Munich
{ala.mellouli, rahul.chaudhari}@tum.de

Abstract

Conventional stories for children are static, independent of the medium (text, video, audio). We aim to make stories interactive by giving the user control over characters, objects, scenes, and timing in known story worlds. This leads to the construction of unique and personalized stories situated in familiar environments. We limit the scope to domains consisting of a coherent body of works, such as the children's book series "Peter Rabbit". We build a knowledge representation for the domain in the form of a character-centric ontology (the symbolic part), and learn that representation automatically. Then, aided by instruction-tuned Large Language Models, we infer a novel storyline over that representation with active user interaction (the neural part). Automated experiments and a human evaluation show that the narratives generated are coherent, enjoyable, and of good quality.

Introduction

Stories are popular among people of all ages, and humans are exceptionally talented at creating them. They are used for entertainment, sharing experiences with others, conveying one's feelings, and strengthening relationships. Although a very challenging topic, Automatic Story Generation is therefore considered to be a highly rewarding Natural Language Processing task.

We are still far from having story generators which can compete with trained writers. Our motivation for this paper is to work on storytelling that does not overlap with classical writing. Interactive story generation started in the early 1970s with the program TaleSpin (Meehan 1977). Herein the story's events are generated dynamically, according to the user's input and guided by the context of the story universe, past events, and a logical final goal. This makes every story generated unique and creates a new form of entertainment while reading since the reader actively participates to different degrees in the story-making process.

This ensures that the events of the story generated are happening in a partially familiar environment for the child, including heroes that the child knows and likes. The work in (Bamman, O'Connor, and Smith 2013), Bamman et al. focuses on the importance of characters in stories. They argued that actions in the plot are propelled by the underlying nature of personas. Therefore, they proposed a method to learn personas through the stereotypical actions the characters engage in, the actions done to them, and the words used to describe them. While we follow similar goals as TaleSpin (Meehan 1977), our method explicitly considers character attributes to constrain the story generation.

In this paper, we first build a character-centric ontology from a coherent body of works such as the popular children's book series "Peter Rabbit". We then leverage instruction-based Large Language Models (LLMs) to generate novel stories that align with this representation and user inputs. Through our story generation system, users can control plot elements and create short children's stories featuring familiar characters. Automated experiments and a human evaluation survey show that the narratives generated are coherent, enjoyable, and of good quality. We call the proposed system "StoReys", hinting at the step-by-step construction of a story on top of previous paragraphs.

Related Work

Neural methods produce original and interesting stories that are well-liked by humans. They require less manual effort than past approaches since they do not rely on predefining the domain knowledge by hand. However, since Language Models such as Recurrent Neural Networks generate text based only on already-seen text tokens, the stories generated are not guaranteed to be coherent or to have a clear ending. To tackle this shortcoming, story generation is split into two main parts in (Martin et al. 2018). They convert sentences in the corpus into more generalized events, represented in a specific format, by replacing words and verbs with their semantic classes. Then, they train an event-to-event predictor to sample an event based on the previous one. Finally, they reconvert this event back to natural language using a Language Model (LM).

The work in (Tambwekar et al. 2019) relies on reinforcement learning to guide the model toward a predefined goal, using a reward-shaping approach. This approach uses the verbs of the generated sentences to indicate how far we are from the goal, and then rewards or punishes the model accordingly through the loss function.

Other works split the generation into separate steps, where they guide the LM through the plot points of the story. A hierarchical generation system is introduced in (Fan, Lewis, and Dauphin 2018), wherein the model first generated a

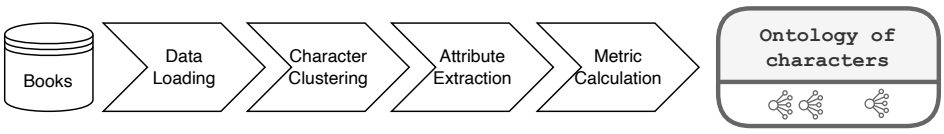

Figure 1: Character representation learning pipeline.

premise sentence of the story and then expands it into a paragraph. The *Plan-And-Write* system from (Yao et al. 2019) takes the title of the story as input and generates keywords that represent a coherent succession of events. It then writes a story following these keywords. The *PlotMachines* program in (Rashkin et al. 2020) proposes an outline-conditioned story generation, in which the user provides sentences that describe key events and characters. The system then automatically orders these plot points and generates a story based on them.

Another way to condition the LM is by providing a beginning and an end to the story. The model then tries to fill in the missing parts based on these two. Narrative interpolation is used in (Wang, Durrett, and Erk 2020), where the model incrementally generates steps based on the previous and the next sentence. Similar to the *Plan-And-Write* system, a hierarchical story-infilling system is proposed in (Ippolito et al. 2019), that generates keywords based on the beginning and end. In a second step, these words are used to condition the LM and then generate the middle part of the story.

Unlike previous studies that primarily focus on text generation using sequence-to-sequence or smaller transformer models, our work investigates instruction-tuned Large Language Models (LLMs) to control plot progression. Since such models can also capture the logical dependencies in the story, an event validation framework is redundant. So we focus instead on building a knowledge representation of the story world that revolves entirely around characters.

To filter out illogical events, the character-centric *CAST* system relies on a commonsense framework; *COMET*, an automatic knowledge inference model that can generate a commonsense description for a given unstructured text (Bosselut et al. 2019). For every generated sentence candidate, this system chooses a main character, infers its intention, and then checks if it matches the previous character's motives. If so, the candidate is deemed to be valid (Peng et al. 2022). In our approach, we rank two candidate paragraphs by comparing them based on the established character-centric knowledge representation, and not on the previous text. This paragraph-based selection generates a story continuation that is more consistent with the characters' personalities.

In (Liu et al. 2020), a character representation is learned first using the embeddings of verbs and adjectives related to each character. They then use this representation to condition an action predictor that decides the character's reaction to the current environment. A story continuation is generated next based on that decision. Similar to this work, our representation is based on verbs and adjectives mentioned for each persona. However, instead of using the words' embeddings, we rely on a commonsense inference method to extract the characters' attributes from unstructured text. This ensures that we capture underlying characteristics that are otherwise not detectable.

Method

We first build a Natural Language Processing pipeline that automatically extracts a character-centric representation from a coherent corpus of stories. We then leverage instruction-based LLMs to iteratively write the story paragraph-by-paragraph. We also incorporate constraints during the sampling phase in order to generate narratives that fulfill the objectives below.

- **Character Faithfulness**: The generated stories should present characters that are similar in personality, actions, and story role, to the character representations extracted from the corpus.
- **Interactivity**: It should be possible to control different elements of the story, with a focus on character-character and character-object interactions.
- **Coherence**: The stories generated should be complete and coherent across their length.

A formal definition of the inputs and output of our method is as follows:

- **Inputs:** Character representations $R = \{r_1, r_2, ..., r_m\}$ are learned from a corpus. Here, r_i is the representation of the i-th character and m is the number of the characters. The user instructions $U = \{u_1, u_2, ..., u_{n-1}\}$ are given to our pipeline to guide the writing. Here, u_i is the user input that constrains the i-th paragraph of the story and n is the total number of paragraphs. We omit the user command u_n since the ending of the story is generated automatically.
- **Output:** A story $P = \{p_0, p_1, ..., p_n\}$ is generated as a result, where p_i is the i-th paragraph of the story.

Character Representation Learning

The sequence of components in the pipeline for learning character representations is shown in Figure 1. First, we retrieve the preprocessed sentences from the stories dataset and cluster them by character. This clustering leverages the Named Entity Recognition (NER) algorithm by *spaCy* (Honnibal et al. 2020), which relies on a word embedding strategy based on sub-word features and a neural network to detect named entities. The next step is to simplify each sentence into clauses to facilitate the attribute inference. We rely on the algorithm *Sentence-to-Clause*[1], which

[1]https://github.com/rahulkg31/sentence-to-clauses

is a lightweight approach that extracts clauses from complex sentences based on dependency trees. This is often prone to errors because phrase constructions are diverse and hard to delimit by simple rules. For this reason, we employ *LanguageTool*[2], to minimize error in the extracted clauses.

From each clause, we then generate a possible character attribute using commonsense inference. In this context, "commonsense" refers to the shared human knowledge of how the world works. It captures information about well-known facts or relationships that a human may infer from natural text. Generally, models are trained on Commonsense Knowledge Graphs (CKGs) and then used to infer commonsense from unseen natural text.

We explore *ATOMIC* (Hwang et al. 2021), a causal knowledge representation where textual descriptions of nine different types of if-then relations are provided. One of these relationships is *xAttr,* which is used for predicting the attribute that describes the subject of a sentence. An example: "*PersonX accepts PersonY's apology.*" One *xAttr* entry in the dataset, in this case, would be the adjective *"forgiving"*. We rely on a pretrained model released together with the dataset. These models are capable of generating accurate commonsense inferences for unseen events. In our work, we adopt the BART-based transformer model (Lewis et al. 2020).

Once the attributes are extracted, the next step is to transform the frequency of each attribute within each character cluster into a more suitable metric.

Metric The representation above is learned by extracting character attributes from all sentences. This extracts multiple personality traits that are not unique to a specific character. Such attributes are too common and should therefore have a lower weight. For this, we propose to use *Term Frequency-Inverse Document Frequency (TF-IDF)*, a numerical statistic commonly used in information retrieval to evaluate how relevant a term is to a specific document in a corpus (Leskovec, Rajaraman, and Ullman 2014). The TF-IDF value increases proportionally to the number of times a term *t* appears in the document *d*. But it is also offset by the number of documents in the corpus *D* that contain the term *t*, which helps to adjust for the fact that some terms appear more frequently in general.

We adapt the TF-IDF nomenclature for our purpose. In this context, we substitute term *t*, document *d*, and corpus *D* in TF-IDF with attribute *a*, character *c*, and set of characters *C*. The TF-IDF is calculated as:

$$tf\_idf(a, c, C) = tf(a, c) \cdot idf(a, C), \tag{1}$$

with

$$tf(a, c) = \frac{n_{a,c}}{\sum_{a' \in c} n_{a',c}}, \tag{2}$$

where $n_{a,c}$ is the number of occurrences of the attribute a for the character c, the denominator is the total number of attributes extracted for character c, by counting each occurrence of the attribute a' separately,

and with

$$idf(a, C) = \log \frac{1 + |C|}{1 + |\{c \in C : a \in c\}|} + 1, \tag{3}$$

[2]www.languagetool.org

Table 1: Comparison table of user interaction and the corresponding real-world motions commonly occuring in children's play with toys.

Web-based UI	**Interaction simulated**
User introduces a new unseen character.	Child brings a new character figurine into the scene.
User introduces a new unseen object.	Child brings a new object into the scene.
User changes the location.	Child changes a location placeholder (such as a card).
User introduces a character movement.	Child moves the figurine of a character.
User introduces a character-character or a character-object interaction.	Child makes two figurines of a character and a character/object interact.

where the numerator is the total number of characters considered. The denominator is the smoothed total number of characters that have the attribute *a*.

Story generation

We want the generation of the story to be user-interactive. In the future, we want to drive the story with inputs from a camera that tracks the motion of a child playing with figurines. This would allow the child to control the story by moving the figurines around. For now, we simulate inputs through a web-based user interface. The side-by-side comparison in the Table 1 illustrates this idea.

Adding Story Elements Users can add characters to the story, and thus introduce roles (main characters, support characters or villain characters) into the narrative. This allows for the development of engaging character dynamics and the progression of the plot. Users can also incorporate objects, which serve as plot devices. Moreover, users can change the location within the story, which provides a shift in the story environment.

Adding Actions Here, users can introduce an interaction between characters or between a character and an object. This enables the creation of meaningful relationships, conflicts, or collaborations within the story. Additionally, users can specify attributes such as speed (fast or slow) and direction (towards, away, or with contact). These attributes add depth to the interactions and describe the motion of the child while playing. Similarly, users can specify movement for a character and define an orientation change (upwards, downwards) or speed (fast or slow) of this motion.

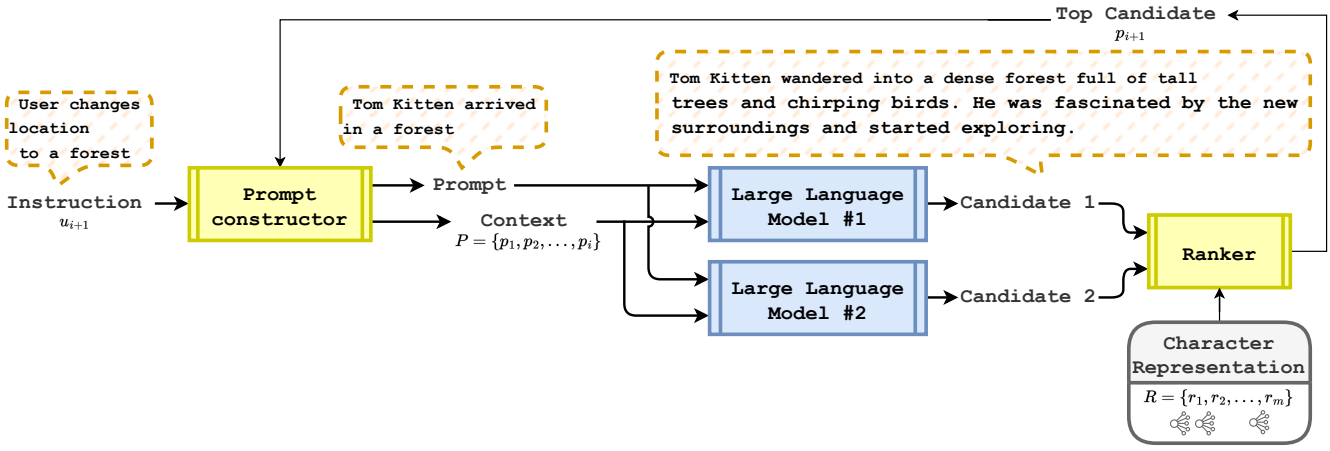

Figure 2: Overview of the story generation pipeline. The user instruction u_{i+1} is translated into a suitable prompt. The story history $\{p_0, p_1, ..., p_i\}$ and the created prompt condition the LLMs. The two LLMs each generate a candidate continuation. A Ranker module compares the candidates and chooses the best one as the story continuation p_{i+1}. The above steps are repeated until a full story $P = \{p_0, p_1, ..., p_n\}$ is generated.

An overview of the story generation pipeline is shown in Figure 2. Two components of the pipeline that we contribute, the *Prompt Construction* and the *Ranking-based Text Generation*, are described next.

Prompt Construction Prompt construction represents a crucial step since it is used for guiding the LLMs, and ensuring that users' intentions as represented by their inputs are fulfilled. Based on user interaction, we automatically generate a suitable prompt that consists of two main parts: a base prompt and an instruction prompt. The base prompt is constant, and it controls the writing style of the story by ensuring a narrative style that is more focused on action and that avoids dialogue. The instruction prompt controls the direction of the story. It specifies if a beginning, a continuation, or an ending is generated. It guarantees that the beginning is randomized and that the story ends on a positive note, or with a moral lesson. It also ensures the motion-based interactivity aspect.

The simulated motion of a child playing with figurines should be transformed into a concrete action that fits both the characteristics of the movement and the role of the characters involved. As an example, if hypothetically a child slowly moves a villain's figurine towards the hero's figurine, it can be interpreted as the villain following the hero or sneaking up on him. To simulate this on the web-based UI, the user would make the following selection: {active character: villain, passive character: hero, action type: movement, direction: towards, modifier: slow}.

We collected a dataset of action verbs for the purpose of sampling an appropriate action for a specific motion. We rely on the language database *FrameNet* to ensure accurate verb semantics (Baker, Fillmore, and Lowe 1998). *FrameNet* is a lexical database of English verbs and nouns, where each word is classified under a semantic frame. As an example, the verb *to help* and the noun *aid* are found under the frame *Assistance*. We manually annotated the selected verbs with a set of selected features. First, each verb specifies the action type it represents. This could include interactions between characters and characters, characters and objects, or character movements. We include possible character roles as an active agent (performing the action) or a passive agent (affected by the action). These roles include main, villain, and support characters. Finally, we added other modifiers such as direction of the movement, speed of the verb motion or changes in orientation.

This results in 120 fully annotated verbs with 53 different semantic frames. These frames cover a considerable spectrum of events. For instance, typical frames for verbs with a villain as an active agent are *Attack*, *Hostile_encounter*, *Inhibit_movement*, among others. Around 67% of the verbs' action type is an interaction, and the remaining 33% represents a movement. During the text generation, based on the action selected and its characteristics and agents, we select a list of appropriate verbs from the dataset. We then extract all unique semantic frames, and randomly choose one. The next step is to then sample one of the verbs that are contained in this frame from the candidate list. Once the verb is selected, the last step is to build a natural language sentence with this verb, which represents the instruction prompt.

Ranking-based Text Generation The text generation pipeline is composed of two main phases: a writing phase and a ranking phase. During the writing phase, as shown in Figure 2, the previously created prompt, coupled with the story history, are fed to multiple LLMs. Based on that, the pretrained models generate two different text candidates as a story continuation. This ensures story diversity, more creative writing and avoids repetition.

The ranking phase is presented in Figure 3. We calculate multiple metrics for each candidate. First, the character score reflects how faithful the candidate is to our character-centric representation. We find and resolve coreferences of the same entities found in the text, using a modified ap-

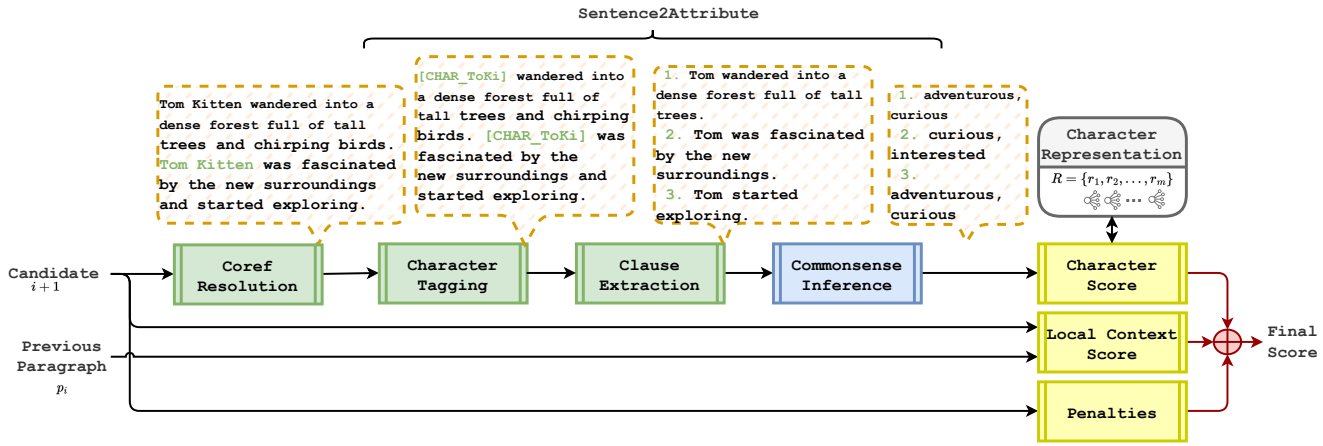

Figure 3: Overview of the candidate ranking pipeline.

proach developed by *NeuroSYS*[3] and based on a coreference resolution model provided by the *AllenNLP* library. We then use the character representation learning pipeline (Figure 1) to extract attributes from the text candidate. These extracted features are matched against the character representation from the entire corpus to calculate a final character score.

Furthermore, we calculate a local context score, which reflects the similarity of the text candidate to the previous paragraph in the story. This ensures that the story appears to be a continuation of the previous events. We transform the candidate and the previous paragraph to a fixed-length vector by leveraging a sentence transformer model called *all-MiniLM-L6-v2*[4]. We then calculate the score based on the *cosine similarity* between these two vectors.

We create the final score by combining the *character score* (70%) and the *local context score* (30%) together using a weighted average. A penalty is additionally applied to this score based on a combination of readability metrics (*flesch reading-ease* score (Flesch 1979), *SMOG* score (Laughlin 1969) and *Dale Chall* score (Dale and Chall 1948)). We reduce the score of a text that is rated higher than a 4th-grade level.

Finally, of the two generated text candidates, the one with the higher score is shown to the user, and added to the memory of the current story paragraphs.

Experiments

Dataset

We select the book series *Peter Rabbit* by *Beatrix Potter* since it contains multiple stories, presents a closed world with recurrent characters, and addresses children between four and eight years old. The books are classic bedtime stories that revolve around animal characters in the countryside, and are freely available online in the digital library *Project Gutenberg*[5]. We only selected the books relevant to eight recurrent characters and removed all dialogue and related sentences that contain speech verbs. We also deleted special punctuation, applied case normalization, and split the text into separate sentences. For that, we use the *Sentence Tokenizer* from the natural language toolkit *NLTK*[6]. This results in a dataset containing 745 sentences with an average length of around 15 words per sentence.

Experiment Setup

We evaluate our approach using two LLMs. Here, the specific choice of LLMs is of no consequence, as our contributions are built around the LLMs and not inside them. The first one is the assistant-style transformer model *GPT-3.5-turbo*, released by *OpenAI*. It was trained on more than 570 GB of text data, coupled with advanced deep learning techniques, such as unsupervised pretraining and reinforcement learning from human feedback. We used the version *gpt-3.5-turbo-0613* through the chat completion API with a temperature of 0.75. The second model is MPT-7B-Chat (MosaicML-NLP-Team 2023), an open-source chatbot-like model for dialogue generation. It was built by finetuning MPT-7B, a decoder-style transformer, using performance-optimized layers and an Attention with Linear Biases (ALiBi) training (Press, Smith, and Lewis 2022). We run a *ggml* format of this model through the *gpt4all* framework. This format makes use of quantization, which allows for large models to run on consumer hardware (Frantar et al. 2022).

Evaluation

Automatic Evaluation We present the following automated metrics used for analyzing the generated stories.

BLEU Score BLEU refers to BiLingual Evaluation Understudy and is a metric used to evaluate the quality of machine-generated text (Papineni et al. 2002). As the name

[3]https://github.com/NeuroSYS-pl/coreference-resolution

[4]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[5]https://www.gutenberg.org/ebooks/14838

[6]https://www.nltk.org/

suggests, this metric is used to evaluate the similarity between a generated text and a ground-truth text by comparing their n-grams, usually for machine translation tasks. It ranges from zero to one, with one indicating that the text overlaps entirely with the ground-truth, and zero meaning that there is no similarity between the two. We compare the generated corpus with the *Peter Rabbit* books as ground truth.

Self-BLEU Score Self-BLEU uses the BLEU score and was first proposed by Zhu et al. (Zhu et al. 2018). Instead of comparing the generated text to the ground-truth text, it compares different parts of the generated text against each other. Similarly, Self-BLEU ranges from zero to one with a lower score indicating a more diverse text. On the other hand, a higher score suggests a repetitive output. This helps assess the variation in a model's generated outputs.

Human Evaluation The previously mentioned metrics do not evaluate the quality of the stories, their coherence, or their novelties. They only evaluate the ability of the model to reproduce a given corpus, which can be useful in some cases but not for story generation. Therefore, the consensus is that human evaluation is currently the only valid method to evaluate text quality in story generation (See et al. 2019). Purdy et al. define a list of survey statements in (Purdy et al. 2018), specifically designed to evaluate generated stories that have been validated against human judgments. We extend this framework by adding additional statements to create a more adequate survey for this work.

We then enlisted 21 subjects to take the survey. To qualify, participants had to read the first book of the *Peter Rabbit* series and a small overview of the recurrent characters. This was provided to the participants in the form of a survey protocol. Participants were then asked to generate two to three short stories through the web-based user interface. In total, 55 stories were generated and evaluated during this survey. To ensure a fixed length of the story, the user was limited to five interactions per story. For each story generation, the participant was asked to truthfully respond to 20 statements using a five-point Likert scale, ranging from one for *Strongly Disagree* to five for *Strongly Agree.*
These statements consist of:

Paragraph-related statements First, for every generated paragraph, we ask the two following questions:

1. (Interactivity) This paragraph's events reflect the character/object's movement in the scene as specified by the user.
2. (Coherence) This paragraph's events make sense given the events before them.

The first question captures the ability of the user to control the various elements of the story. The second question focuses on the coherence of the generated text. This demonstrates if the LLM can generate events that are logical and coherent with the previous events of the story.

Table 2: BLEU n-gram (BLEU-n) scores between 700 generated story sentences of each model and the *Peter Rabbit* dataset. A higher BLEU score indicates more similarity to the books, and we highlight the highest model score in each column.

	BLEU-2	BLEU-3	BLEU-4	BLEU-5
MPT	**0.504**	**0.222**	**0.116**	**0.070**
GPT3.5	0.379	0.151	0.078	0.050
Average	0.442	0.187	0.097	0.060

Story-related statements The statements in this section are about the story as a whole:

1. This story's events occur in a plausible order.
2. This story avoids repetition.
3. This story uses interesting language.
4. This story is enjoyable.
5. This story reminds me of a Peter Rabbit story.
6. The characters appeared to be consistent across this story.
7. The characters in this story have personalities similar to the Peter Rabbit characters.

Questions (1. - 4.) were taken from the list of survey statements designed in (Purdy et al. 2018). Further two questions (5. and 7.) assess the similarity of the story generated and the characters' traits to the *Peter Rabbit* world. An additional question (6.) was included to explore whether the characters' roles and personalities were conserved during the story generation.

AI-related statements Finally, we also include statements about the cooperation between the participant and the AI used in this system. These are listed below:

1. I can count on the story generation system to write a reliable story.
2. I felt engaged in the story writing.
3. The story generation system leaves enough room for my creativity.
4. Overall, I am satisfied with the story generation system in this scenario.

The first question was added to investigate if the user trusts the system to write a good story. With the second and third statements, we investigate the user's engagement and creativity during the generation process. The last statement is concerned with the overall satisfaction of the participant with the AI-system.

Results and Discussion

Automatic Evaluation

Similarity For the BLEU score, the results are shown in Table 2. We first interpret the average results of both story writers together. Nearly 44% of the bigrams in the generated text are found in the books. This shows a moderate similarity between the generated and reference text for bigrams. As

Table 3: Self-BLEU n-gram (Self-BLEU-n) scores of 700 generated story sentences of each model and the ground truth dataset sentences (*Peter Rabbit* books). A lower self-BLEU (denoted as S-B in the table below) score indicates more diversity. We highlight the lowest model score in each column (other than the *Ground Truth Dataset*).

	S-B-2	S-B-3	S-B-4	S-B-5
Ground truth Dataset	0.637	0.364	0.199	0.124
MPT	0.811	0.658	0.537	0.454
GPT3.5	**0.718**	**0.512**	**0.366**	**0.274**

for the BLEU-3 score, the generated text has fewer trigrams (ca. 18%) in common with the books. As for 4-grams and 5-grams, the scores suggest even less overlap (ca. 9% of 4-grams and ca. 6% of 5-grams). This could indicate a considerable difference between the generated stories and the *Peter Rabbit* stories. However, this is only a weak claim as BLEU-n score is expected to generally decrease as n grows.

From Figures 4 and 5, we observe that the high bigram score reflects mostly characters' names and common character-verb combinations. The low BLEU scores also highlight that the story writer does not aim to replicate the *Peter Rabbit* books. We also observe that the MPT model generates text that is more similar to the *Peter Rabbit* books than the GPT3.5 model.

Diversity Table 3 presents the self-BLEU scores for both the generated stories for each model and the relevant books (ground truth dataset). To calculate this score, we treat one sentence as the hypothesis and the others as references. This process is repeated for every sentence of a corpus of 700 generated sentences, and the average BLEU score is considered the self-BLEU score.

The story writer based on GPT3.5 model performed well in the diversity score, with self-BLEU scores slightly higher but extremely similar to the original stories. This indicates that the language diversity of the model aligns with human-written text and avoids repetition. As for the MPT model, higher self-BLEU scores show that the generations are more repetitive and less diverse.

Human Evaluation

The overall results of the survey are shown in Figure 6, where we detail the percentages of the Likert scores for each dimension in the questionnaire. Generally, participants responded positively to our story generation system. Across all dimensions, positive Likert scores range from 60% to 95%, while negative scores are between 5% and 40%.

Before discussing the results, we first validate the conducted evaluation using the metric *Cronbach's Alpha* α, which is used to assess the internal consistency of a survey. This statistic measures how well the survey items are correlated and consistently target the same underlying concept. It ranges between zero and one, with higher values indicating stronger internal reliability, while a lower value suggests poorer consistency. For our survey, $\alpha = 0.877$, signifying a very good internal consistency.

Paragraph-related Results We observe that the story writer produces continuations that are coherent and faithful to the user's input. In the context of interactivity, the writer is good at following instructions, especially for the first paragraph, where the context is just a short beginning. Therefore, it is easier for the LLM to generate a quality continuation. For the following story parts and as the context increases in size, the interactivity rating drops slightly.

With the progression of the story, the writer is able to produce contextually relevant text, that aligns well with past story elements. It could remember past interactions and objects that appeared before, even when the story context spanned over multiple paragraphs. However, we noted a slight decline in the coherence of these paragraphs, as the story events exhibit on occasion an increase in randomness. An example would be that the writer sometimes forgets one of the characters that appeared at the very start of the story, or that the writer fails to develop an important plot point and

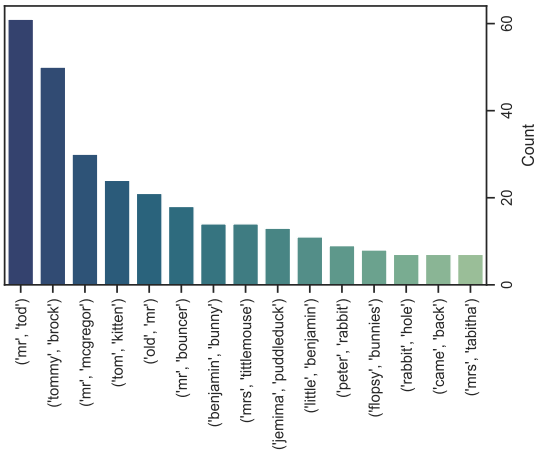

Figure 4: Bigrams in the original *Peter Rabbit* books.

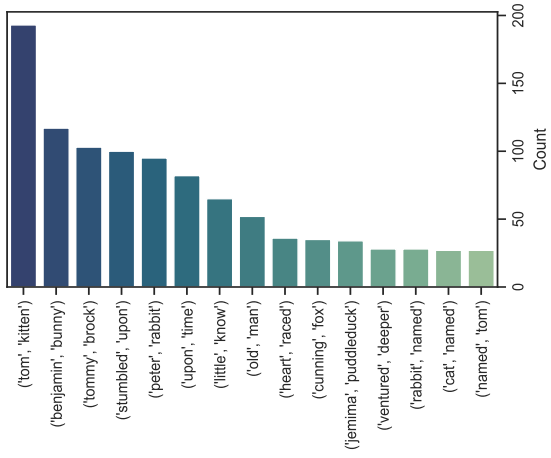

Figure 5: Bigrams in the generated *Peter Rabbit* stories.

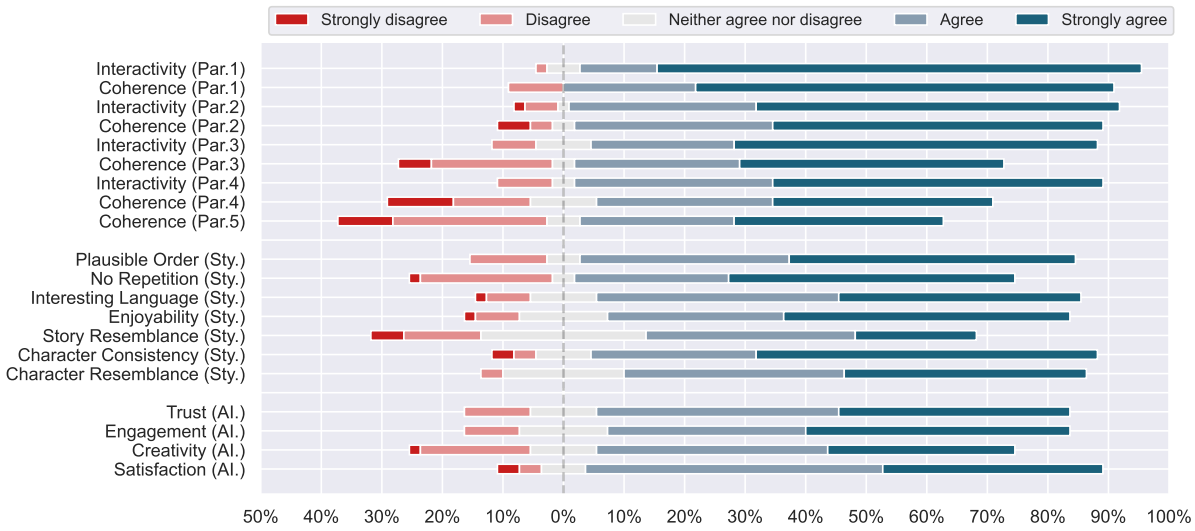

Figure 6: Distribution of the Likert score results in percentages per survey statement. "Par.i" stands for paragraph-related results for paragaph i, "Sty." for story-related results, and "AI." for AI-related results.

rushes through it.

Generally, the writer succeeded in generating text that incorporates new elements into the story. Whether it is changing the story setting to another location or adding a villain, the system allows the participants to control the plot direction, while still producing coherent stories.

While user feedback for both interactivity and coherence is generally positive, we observed a small drop across both metrics when adding an action. This can be explained by the fact that the verb sampled from the verbs dataset can occasionally effect an abrupt change in the course of the story. This, at times, leads to the generation of events that lack coherence.

Story-related Results Our system generates events in a logical order. It also uses interesting language and avoids repetition, which makes the stories more enjoyable for the participants. Additionally, the characters in the stories appear consistent, i. e., the characters' personality traits and roles do not change drastically throughout the story. Next, we studied the resemblance to the original books. We observed that the participants were neutral about this statement since the writing style of the original books uses an older style of English writing, which set them apart from the generated stories.

On the other hand, the feedback for character resemblance to the original books is more positive. The character-based ranking, coupled with prompting and the capabilities of the LLM, made it possible to produce characters that are similar to the characters of the *Peter Rabbit* world. We observed that the stories included characteristic actions, such as the character *Jemima Puddle-duck* wanting to lay her eggs, which was the main plot point of one of the original books.

AI-related Results We then study the features related to human-AI cooperation. We note that the feedback across the four statements is generally positive, with an average of around four. The participants were found to trust the system to generate the stories, and are satisfied with the end result. The participants were engaged in the generation process due to the interactivity aspect, and the writer produced text that promotes the user's creativity.

Conclusion

We proposed a story generator that automatically extracts character attributes from an unstructured story corpus. These attributes are then clustered and transformed into a suitable metric. During story generation, text prompts aligning with user input are constructed using an annotated verbs list. Two instruction-tuned LLMs were used to produce text candidates. These candidates were then evaluated with a ranking module that relies on the knowledge representation and additional metrics to ensure a good quality of story continuation. This results in novel action-oriented narratives. Users of the system were able to control plot elements and to generate short children's stories that include characters similar to the original personas. Based on a set of automated experiments and a human survey, we conclude that the stories generated are coherent, enjoyable, and of good quality.

Author Contributions

Alaeddine developed the story generation tool, conducted the user studies, and analyzed the data. Rahul defined the topic and supervised the project. Alaeddine wrote the first paper draft. Rahul reviewed it and made the final revisions.

Acknowledgments

Wilder Lopes facilitated the development of the story generation tool by kindly providing access to the LLM API.

References

[Baker, Fillmore, and Lowe 1998] Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, 86–90. Montreal, Quebec, Canada: Association for Computational Linguistics.

[Bamman, O'Connor, and Smith 2013] Bamman, D.; O'Connor, B.; and Smith, N. A. 2013. Learning latent personas of film characters. In Schuetze, H.; Fung, P.; and Poesio, M., eds., *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 352–361. Sofia, Bulgaria: Association for Computational Linguistics.

[Bosselut et al. 2019] Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Celikyilmaz, A.; and Choi, Y. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In Korhonen, A.; Traum, D.; and Màrquez, L., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4762–4779. Florence, Italy: Association for Computational Linguistics.

[Dale and Chall 1948] Dale, E., and Chall, J. S. 1948. A formula for predicting readability. *Educational Research Bulletin* 27(1):11–28.

[Fan, Lewis, and Dauphin 2018] Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 889–898. Melbourne, Australia: Association for Computational Linguistics.

[Flesch 1979] Flesch, R. 1979. *How to Write Plain English: A Book for Lawyers and Consumers*. Harper & Row.

[Frantar et al. 2022] Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*.

[Honnibal et al. 2020] Honnibal, M.; Montani, I.; Van Landeghem, S.; and Boyd, A. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

[Hwang et al. 2021] Hwang, J. D.; Bhagavatula, C.; Bras, R. L.; Da, J.; Sakaguchi, K.; Bosselut, A.; and Choi, Y. 2021. COMET-ATOMIC 2020: On symbolic and neural commonsense knowledge graphs.

[Ippolito et al. 2019] Ippolito, D.; Grangier, D.; Callison-Burch, C.; and Eck, D. 2019. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, 37–43. Minneapolis, Minnesota: Association for Computational Linguistics.

[Laughlin 1969] Laughlin, G. H. M. 1969. Smog grading-a new readability formula. *Journal of Reading* 12(8):639–646.

[Leskovec, Rajaraman, and Ullman 2014] Leskovec, J.; Rajaraman, A.; and Ullman, J. D. 2014. *Mining of Massive Datasets*. USA: Cambridge University Press, 2nd edition.

[Lewis et al. 2020] Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. Online: Association for Computational Linguistics.

[Liu et al. 2020] Liu, D.; Li, J.; Yu, M.-H.; Huang, Z.; Liu, G.; Zhao, D.; and Yan, R. 2020. A character-centric neural model for automated story generation. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(02):1725–1732.

[Martin et al. 2018] Martin, L. J.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in AI*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

[Meehan 1977] Meehan, J. R. 1977. Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, 91–98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[MosaicML-NLP-Team 2023] MosaicML-NLP-Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms.

[Papineni et al. 2002] Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 311–318. USA: Association for Computational Linguistics.

[Peng et al. 2022] Peng, X.; Li, S.; Wiegreffe, S.; and Riedl, M. 2022. Inferring the reader: Guiding automated story generation with commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 7008–7029. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

[Press, Smith, and Lewis 2022] Press, O.; Smith, N. A.; and Lewis, M. 2022. Train short, test long: Attention with linear biases enables input length extrapolation.

[Purdy et al. 2018] Purdy, C.; Wang, X.; He, L.; and Riedl, M. 2018. Predicting generated story quality with quantitative measures. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 14(1):95–101.

[Rashkin et al. 2020] Rashkin, H.; Celikyilmaz, A.; Choi, Y.; and Gao, J. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. In *Conf. Empirical Methods in Natural Language Processing*, 4274–4295.

[See et al. 2019] See, A.; Pappu, A.; Saxena, R.; Yerukola, A.; and Manning, C. D. 2019. Do massively pretrained language models make better storytellers? In Bansal, M., and Villavicencio, A., eds., *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*,

843–861. Hong Kong, China: Association for Computational Linguistics.

[Tambwekar et al. 2019] Tambwekar, P.; Dhuliawala, M.; Martin, L. J.; Mehta, A.; Harrison, B.; and Riedl, M. O. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 5982–5988. International Joint Conferences on Artificial Intelligence Organization.

[Wang, Durrett, and Erk 2020] Wang, S.; Durrett, G.; and Erk, K. 2020. Narrative interpolation for generating and understanding stories. *ArXiv* abs/2008.07466.

[Yao et al. 2019] Yao, L.; Peng, N.; Weischedel, R.; Knight, K.; Zhao, D.; and Yan, R. 2019. Plan-And-Write: towards better automatic storytelling. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

[Zhu et al. 2018] Zhu, Y.; Lu, S.; Zheng, L.; Guo, J.; Zhang, W.; Wang, J.; and Yu, Y. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, 1097–1100. New York, NY, USA: Association for Computing Machinery.

Appendix

Example generated story

In the heart of a dense forest, Benjamin Bunny, a curious rabbit with soft brown fur, hopped through the vibrant undergrowth. Sunlight danced through the treetops, casting playful shadows on the forest floor. As Benjamin explored, he stumbled upon an ancient oak tree, its gnarled branches reaching toward the heavens.

(Add character [Mr. Tod]) As Benjamin ventured closer to the ancient oak, a shadowy figure emerged from the depths of the forest. It was Mr. Tod, a cunning and sly fox known for his mischief. With a wicked grin, Mr. Tod locked eyes with Benjamin, his sharp teeth glinting in the dappled light. The rabbit's heart raced as he realized he had stumbled into the clutches of a notorious villain.

(Move [Mr. Tod] fast towards [Peter Rabbit]) Mr. Tod lunged forward, his jaws snapping inches away from Benjamin's fluffy tail. The nimble rabbit, quick on his paws, darted away, zigzagging through the forest. Branches swiped at his fur as he led Mr. Tod on a wild chase. Benjamin's heart pounded in his chest, but determination fueled his tiny legs. He had to outsmart the cunning fox and find a way to escape his clutches.

(Change location to a [Cave]) With Mr. Tod hot on his heels, Benjamin spotted a hidden cave nestled amidst the ancient trees. Without hesitation, he darted inside, the darkness enveloping him. The cave offered a temporary refuge, but Benjamin knew he couldn't stay for long. As his eyes adjusted, he noticed a narrow tunnel leading deeper into the unknown.

(Move [Mr. Tod] slowly towards [Peter Rabbit]) The narrow tunnel led Benjamin Bunny to a hidden exit, revealing a vast farm sprawling before him. As he cautiously explored, the scent of crops filled the air. But to his dismay, he realized Mr. Tod had followed him. Benjamin hopped across the fields, seeking shelter amidst the haystacks and barns, determined to outwit the fox once and for all in this unfamiliar territory.

As Benjamin hid behind a haystack, he heard a commotion. Curiosity compelled him to peek out and his eyes widened in amazement. Farmer Brown had arrived, scaring Mr. Tod away. Grateful, Benjamin hopped towards Farmer Brown, who smiled kindly. From that day on, Benjamin discovered the joy of friendship and learned that sometimes, unexpected allies can come to the rescue. And so, in the farm's safety, Benjamin Bunny found a new home and a forever friend.