LyricJam: A system for generating lyrics for live instrumental music

Olga Vechtomova, Gaurav Sahu, Dhruv Kumar
University of Waterloo
{ovechtom, gsahu, d35kumar}@uwaterloo.ca

Abstract

We describe a real-time system that receives a live audio stream from a jam session and generates lyric lines that are congruent with the live music being played. Two novel approaches are proposed to align the learned latent spaces of audio and text representations that allow the system to generate novel lyric lines matching live instrumental music. One approach is based on adversarial alignment of latent representations of audio and lyrics, while the other approach learns to transfer the topology from the music latent space to the lyric latent space. A user study with music artists using the system showed that the system was useful not only in lyric composition, but also encouraged the artists to improvise and find new musical expressions. Another user study demonstrated that users preferred the lines generated using the proposed methods to the lines generated by a baseline model.

Introduction

Music artists have different approaches to writing song lyrics. Some write lyrics first, and then compose music, others start with the music and let the mood and the emotions that emerge from the music guide their choice of lyric themes, words and phrases. The latter type of songwriting is often experimental in nature, and is commonly found in genres such as rock, jazz and electronic music, whereby one or more artists jam and in the process converge on the desired musical expressions and sounds. In this work, we describe a system we designed to support this type of song writing process, where artists play musical instruments, while the system listens to the audio stream and generates new lyric lines that are congruent with the music being played in real time. The artists see the lines as they are generated in real time, potentially entering a feedback loop, where the lines suggest phrases and themes that artists can use to guide their musical expressions and instrumentation as they play their instrument. The generated lines are not intended to be the complete song lyrics, instead acting as snippets of ideas and expressions inspiring the artist's own creativity. After the jam session is over, the lines are saved, and the artist can use them as inspiration to write the lyrics for their song.

Our intent is to design a system capable of generating original lyric lines that match the emotions and moods evoked by live instrumental music. Past research in musicology has found a correlation between some aspects of music and emotions. In one large-scale study, researchers found evidence that certain harmonies have strong associations with specific emotions, for example, the diminished seventh is associated with the feeling of despair (Willimek and Willimek 2014). In another study, researchers found correlation between major chords and lyric words with positive valence (Kolchinsky et al. 2017). In addition to harmonies, various sound textures and effects also contribute to the emotional intent of the music. Therefore, in this work we use raw audio input that captures all aspects of music.

The proposed model is trained in an unsupervised manner, being only shown aligned data, consisting of an audio clip and its corresponding lyric text. No labels have been assigned to the data. During training, the model can learn any discernible associations between the raw audio characteristics captured in the spectrograms and the texts of lyrics. At inference time, the model receives clips sampled from the live audio stream of the music played by the user, and generates new lines. The associations learned by the model determine various characteristics of the generated lines, which could include sentiment and stylistic markers, lexical expressions, syntactic characteristics and syllable patterns.

Our approach is based on training a variational autoencoder for learning the representations of Mel-spectrograms of audio clips (spec-VAE), and a conditional variational autoencoder for learning the representations of lyric lines (text-CVAE). The advantage of using variational autoencoders as generative models is their ability to learn a continuous latent space that can then be sampled to generate novel lines, which is an important requirement for creative applications.

At inference time, the model must be able to generate new lyric lines given an audio clip being recorded from live jam session. In order to do that we need a way to align the latent representations learned by the spec-VAE and the latent representations learned by the text-VAE. We propose two novel approaches to achieve this alignment.

The first approach (Figure 1) is based on training a separate Generative Adversarial Network (GAN) model that takes the spectrogram embedding from spec-VAE, and learns to predict the lyric line embedding in the text-CVAE. The GAN-predicted embedding is then sent to the text-CVAE decoder to generate text.

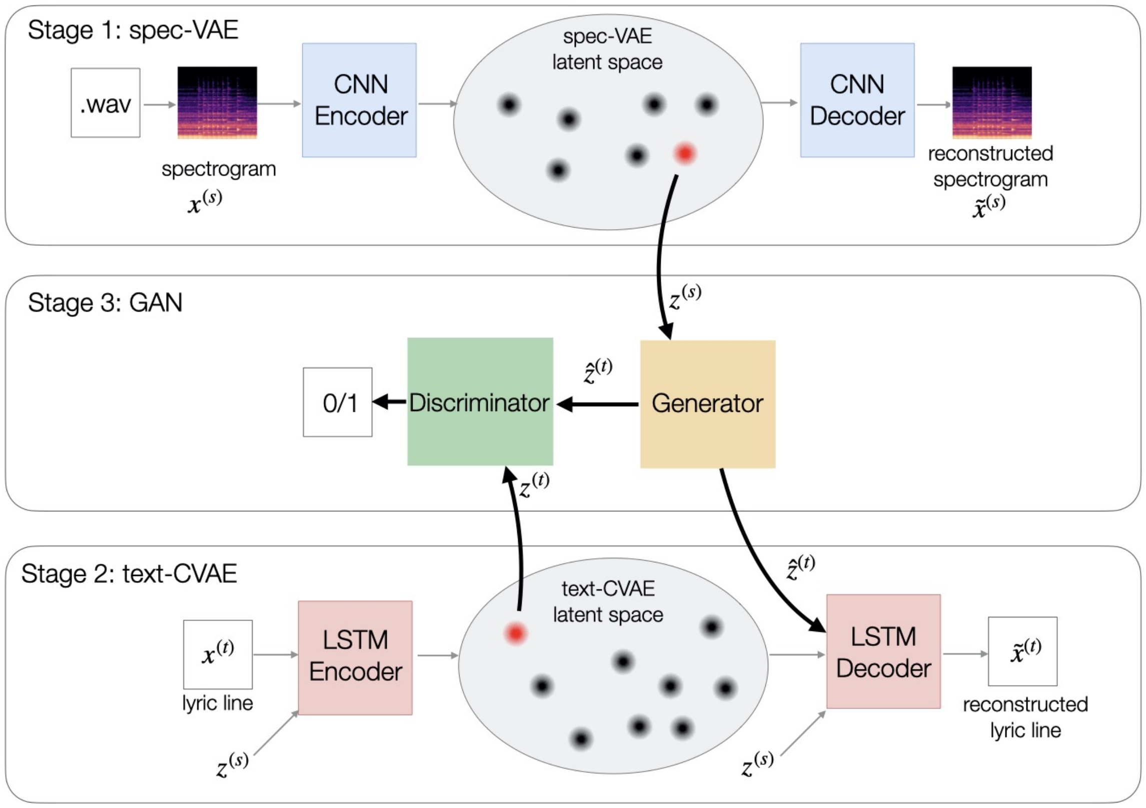


Figure 1: GAN-based alignment of music and lyrics representations (Approach 1).

The second approach (Figure 2) learns to transfer the latent space topology of the spec-VAE to the text-CVAE latent space. To achieve this we use the learned posteriors from the spec-VAE as priors in text-CVAE during its training. The text-CVAE learns to encode lyric lines corresponding to a given audio clip in the region of the latent space corresponding to that clip. Also, since similar sounding audio clips are encoded in neighboring regions, the text-CVAE correspondingly learns to encode lines for these clips in neighboring regions. For example, ambient music clips would be encoded in neighboring regions of spec-VAE, and so would be the lines corresponding to these clips. The intuition is that lines corresponding to similar sounding audio clips (e.g. ambient) would have similar emotional intent, as opposed to, for example, aggressive sounding music. At inference time, when an artist plays an ambient music piece, the system would feed its spectrogram to the spec-VAE encoder to get the parameters of its posterior distribution. Since the spec-VAE posterior distributions are also prior distributions in the text-CVAE, the system samples latent codes from the corresponding prior of the text-CVAE, generating new lines reflecting ambient music.

To summarize, the contributions of this paper are as follows:

1. To our knowledge this is the first real-time system that receives a live audio stream from a jam session and generates lyric lines that are congruent with the live music being played.
2. We propose two novel approaches to align the latent spaces of audio and text representations that allow the system to generate novel lyric lines matching the live music audio clip.
3. We discuss our findings based on observations and interviews of musicians using the system.

In the next section, we describe the GAN-based approach of aligning the spec-VAE and text-CVAE latent embeddings. Next, we will present our second approach of aligning the topologies of the two VAEs. We will then describe user studies and present our findings, which will be followed by the discussion of related work, implementation details and conclusions.

Approach 1: GAN-based alignment of music and lyrics representations

The model consists of three neural network models that are trained consecutively in three stages (Figure 1).

Training stage 1: spec-VAE

In this stage, we train the spectrogram variational autoencoder (VAE) model to learn the latent representations of audio clips.

First we convert the raw waveform audio files into Mel-spectrogram images using the same method as used in Vech-

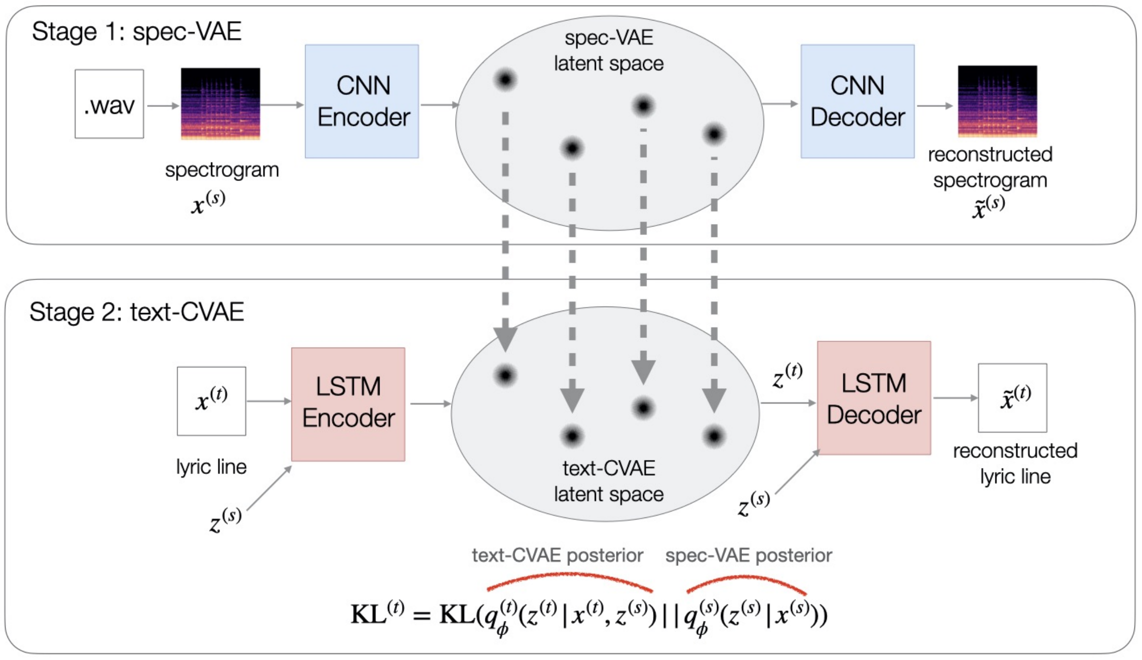


Figure 2: Latent space topology transfer from spec-VAE to text-CVAE (Approach 2).

tomova, Sahu, and Kumar (2020). These spectrograms are then used as input for the spec-VAE.

The variational autoencoder (Kingma and Welling 2014) is a stochastic neural generative model that consists of an encoder-decoder architecture. The encoder transforms the input image $\boldsymbol{x}$ into the approximate posterior distribution $q_\phi(z|x)$ learned by optimizing parameters ϕ of the encoder. The decoder reconstructs $\boldsymbol{x}$ from the latent variable $\boldsymbol{z}$, sampled from $q_\phi(z|x)$. In our implementation, we use convolutional layers as the encoder and a deconvolutional layers as the decoder. Standard normal distribution was used as the prior distribution $p(z)$. The VAE is trained on the loss function that combines the reconstruction loss (Mean Squared Error) and KL divergence loss that regularizes the latent space by pulling the posterior distribution to be close to the prior distribution.

Training stage 2: text-CVAE

Unlike the vanilla VAE used for encoding spectrograms, we use conditional VAE (CVAE) for encoding lyrics.

The CVAE learns a posterior distribution that is conditioned not only on the input data, but also on a class c: $q_\phi(z|x,c)$. Here, we define the class as the spectrogram corresponding to a given line. Every conditional posterior distribution is pulled towards the same prior, here the standard normal distribution.

During training, every input data point consists of a lyric line and its corresponding spectrogram. We first pass the spectrogram through the spec-VAE encoder to get the parameters of the posterior distribution (a vector of means and a vector of standard deviations). We then sample from this posterior to get a vector $z^{(s)}$ that is then concatenated with the input of the encoder and the decoder during training. The reason why we used sampling as opposed to the mean $z^{(s)}$ vector is to induce the text-VAE model to learn conditioning on continuous data, as opposed to discrete classes. This prepares it to better handle conditioning on unseen new spectrograms at inference.

Both the encoder and the decoder in the text-CVAE are Long Short Memory Networks (LSTMs). The sampled $z^{(s)}$ is concatenated with the word embedding input to every step of the encoder and the decoder.

Training stage 3: GAN In this phase, we train a generative adversarial network (GAN), which learns to align audio and text latent codes. The GAN architecture has a generator G and a discriminator D. For training the GAN on a given spectrogram-text pair $\{x^{(s)}, x^{(t)}\}$[1], we follow these steps:

1. First, we pass the spectrogram $x^{(s)}$ through spec-VAE to obtain $z^{(s)} = \mu^{(s)} + \tau(\epsilon \cdot \sigma^{(s)})$, the latent code sampled from the posterior distribution. Here, $\mu^{(s)}$ and $\sigma^{(s)}$ denote the mean and standard deviation predicted by the spec-VAE, $\epsilon \sim \mathcal{N}(0,1)$ is a random normal noise, and τ is the sampling temperature. Simultaneously, we obtain $z^{(t)} = \mu^{(t)} + \tau(\epsilon \cdot \sigma^{(t)})$ by passing the corresponding lyric line $x^{(t)}$ through the text-VAE.
2. After obtaining $z^{(s)}$ and $z^{(t)}$, we proceed with the GAN training. We pass $z^{(s)}$ through the generator network, which outputs a predicted text latent code $\hat{z}^{(t)}$.
3. We then pass $\hat{z} = [\hat{z}^{(t)}; z^{(s)}]$ and $z = [z^{(t)}; z^{(s)}]$ through

[1]Superscripts $^{(s)}$ and $^{(t)}$ in our notation refer to spectrogram and text, respectively

the discriminator network, where ; denotes the concatenation operation. We treat $\hat{z}$ as the negative sample, and z as the positive sample. The discriminator D then tries to distinguish between the two types of inputs. This adversarial training regime, in turn, incentivizes G to match $\hat{z}^{(t)}$ as closely as possible to $z^{(t)}$.

The adversarial loss is formulated as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim \mathcal{D}_{\text{train}}}[\log D(z) + \log(1 - D(\hat{z})] \tag{1}$$

where $\mathcal{D}_{\text{train}}$ is the training data, and each sample $x = \{x^{(s)}, x^{(t)}\}$. We also add an auxiliary MSE loss to the objective function as it is found to stabilize GAN training (Khan et al. 2020). The overall loss for the GAN is:

$$J_{GAN} = \min_G \max_D V(D,G) + \lambda_{MSE} ||\hat{z}^{(t)} - z^{(t)}|| \tag{2}$$

At inference time, the encoder of the text-CVAE is no longer needed. A spectrogram is input to the spec-CVAE encoder to obtain the spectrogram latent code $z^{(s)}$, which is then fed to the generator of the GAN, which generates the lyric latent code $z^{(t)}$. The inference process is also stochastic, as $z^{(s)}$ is sampled from the posterior distribution for s. Sampling allows us to generate diverse lines for the same spectrogram. The GAN-predicted text latent code is then concatenated with the spectrogram latent code and input to the text-CVAE decoder, which generates a lyric line.

Approach 2: Latent space topology transfer from spec-VAE to text-CVAE

The intuition for this approach is to induce the text-CVAE to learn the same latent space topology as the spec-VAE. This would mean that data points that are close in the spec-VAE latent space are expected to be close in the text-VAE latent space. More concretely, if two audio clips are encoded in the neighboring regions of the spec-VAE latent space, their corresponding lyric lines should also be encoded in the neighboring regions in the text-CVAE latent space.

The training is a two-stage process (Figure 2), where the first stage, spec-VAE, is the same as in the GAN-based approach. For the second stage, we train a different formulation of text-CVAE. Instead of using one prior (standard normal) to regularize every posterior distribution, we use the posterior of the spec-VAE as the prior for any given data point. More formally, let the spectrogram be $x^{(s)}$ and its corresponding lyric line be $x^{(t)}$. The posterior distribution for the spectrogram in the spec-VAE is $q_\phi^{(s)}(z^{(s)}|x^{(s)})$, and the posterior distribution for the lyric line in the text-CVAE is $q_\phi^{(t)}(z^{(t)}|x^{(t)}, z^{(s)})$. The KL term of the text-CVAE loss is computed between the posterior for the lyric line and the prior which is set to be the posterior of its corresponding spectrogram in spec-VAE:

$$\mathrm{KL}^{(t)} = \mathrm{KL}(q_\phi^{(t)}(z^{(t)}|x^{(t)}, z^{(s)}) || q_\phi^{(s)}(z^{(s)}|x^{(s)})) \tag{3}$$

The cross-entropy reconstruction loss for text-CVAE is:

$$J_{\text{rec}}(\phi, \theta, z^{(s)}, x^{(t)}) = -\sum_{i=1}^{n} \log p(x^{(t)}|\boldsymbol{z}^{(t)}, z^{(s)}, {x_1}^{(t)} \cdots {x_{i-1}}^{(t)}) \tag{4}$$

The final text-CVAE loss is the combination of the reconstruction loss and the KL term:

$$J_{\text{CVAE}}(\phi, \theta, z^{(s)}, x^{(t)}) = J_{\text{rec}} + \lambda\, \mathrm{KL}^{(t)} \tag{5}$$

To avoid the KL term collapse problem, we used two techniques first proposed by Bowman et al. (2016): KL term annealing by gradually increasing λ and word dropout from the decoder input.

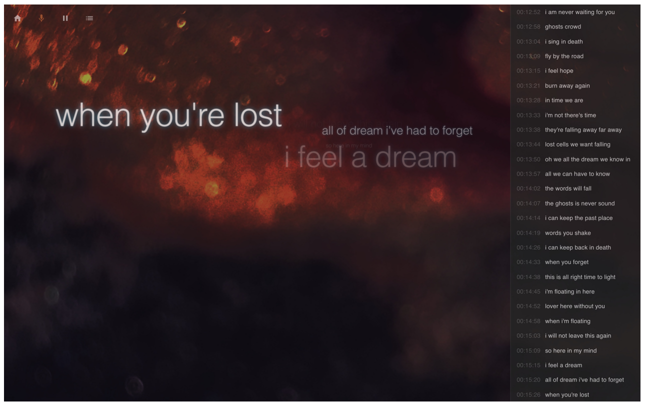
Ranking of generated lines with BERT

At inference time, the model generates a batch of 100 lines conditioned on a short audio clip sampled every 10 seconds from the user's audio source. Since the system shows one or two lines to the user for each clip, and since not all generated lines are equally fluent or interesting, we need a method to rank them so that the system shows a small number of high-quality lines to the user. For this purpose, we used a pre-trained BERT model (Devlin et al. 2019), which we fine-tuned on our custom dataset. The dataset consisted of 3600 high-quality and low-quality lines that were manually selected by one of the authors from a large number of lines output by a VAE trained on the same dataset as used in our experiments. The criteria used for determining quality of lines included the following: originality, creativity, poetic quality and language fluency. While BERT is trained as a binary classifier, we use logits from the final layer for ranking the lines.

Live lyric generation system

We developed a React/NodeJS web application[2] that listens to the user's audio source, which can be either a microphone or line level input from the user's audio-to-digital converter, receiving input from the user's instruments, such as guitar or keyboard. The application samples clips from the audio stream every 10 seconds and saves them as uncompressed PCM WAV files at 44.1kHz sampling rate. On the server, WAV files are converted to Mel-spectrograms and sent to the spec-VAE to obtain the latent code, which is then used in lyric generation by the text-CVAE. The lines generated by text-CVAE are passed through BERT classifier for ranking. The top-ranked lines are then displayed to the user on their screen. The lines slowly float for a few seconds, gradually fading away as newly generated lines appear. The user can view the history of all generated lines with time stamps during the jam session in a collapsible side drawer (Figure 3).

[2]The application can be accessed at: https://lyricjam.ai


when you're lost

all of dream i've had to forget

i feel a dream

```
00:12:52  i am never waiting for you
00:12:58  ghosts crowd
00:13:04  i sing in death
00:13:09  fly by the road
00:13:15  i feel hope
00:13:21  burn away again
00:13:28  in time we are
00:13:33  i'm not there's time
00:13:38  they're falling away far away
00:13:44  lost cells we want falling
00:13:50  oh we all the dream we know in
00:13:57  all we can have to know
00:14:02  the words will fall
00:14:07  the ghosts is never sound
00:14:14  i can keep the past place
00:14:19  words you shake
00:14:26  i can keep back in death
00:14:33  when you forget
00:14:38  this is all right time to light
00:14:45  i'm floating in here
00:14:52  lover here without you
00:14:58  when i'm floating
00:15:03  i will not leave this again
00:15:09  so here in my mind
00:15:15  i feel a dream
00:15:20  all of dream i've had to forget
00:15:26  when you're lost
```

Figure 3: LyricJam screenshot.

Evaluation

We trained the system on the aligned lyric-music dataset (Vechtomova, Sahu, and Kumar 2020), consisting of 18,000 WAV audio clips of original songs and their corresponding lyrics by seven music artists in the Rock genre.

The goals of our evaluation are two-fold:

1. Determine whether the two proposed methods generate lines that match the mood created by the music more accurately.
2. Understand how the application can benefit music artists as they compose new songs by playing musical instruments.

To answer these evaluation questions, we designed two user studies, described in detail in the subsequent sections.

Study 1

For this study, we developed an interactive web application, which plays instrumental songs and every 10 seconds shows three lines to the users in random order, two generated by the proposed models and one by the baseline model. The user was asked to click on the line that best matches the mood of the music currently being played. Participants for this study did not have to be songwriters or music artists, but they were required to have a general appreciation of music of any genre.

For the baseline, we used CVAE with standard normal prior. This is a strong baseline, which generates lines conditioned on the music being played. Comparison to this baseline specifically lets us evaluate the effect of the two proposed methods: GAN-based alignment (GAN-CVAE) and topology transfer (CVAE-spec). At inference time, the lines generated by each system were ranked by BERT, and the top-ranked line from each system was selected for the experiment.

We selected five instrumental songs by two artists for this study with a variety of tempos and instrumentation, and evoking different moods. The lines for each 10-second song clip were generated in advance, so that each user was shown exactly the same three lines per clip. The order of line presentation was random each time. Examples of generated lyrics and the spectrograms of corresponding audio clips can be seen in Figure 4.

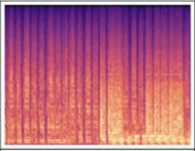
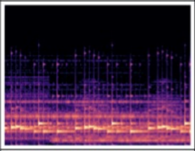
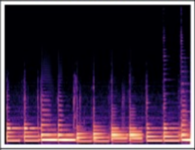
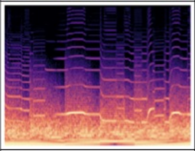
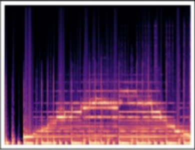
	CVAE: new time to hate CVAE-spec: we're the consequence GAN-CVAE: i want to break my stars Music: harsh synth and crunchy guitar riff
	CVAE: all there is CVAE-spec: I'm running out GAN-CVAE: oh the loneliest dream Music: sombre marimba and dark synth drone
	CVAE: never time to be ... ? CVAE-spec: take in time again GAN-CVAE: after the moon Music: forlorn plucked bass guitar riff
	CVAE: and the sky will never go back CVAE-spec: all i can see GAN-CVAE: shadows in the sun Music: moody layered distorted synth
	CVAE: all the love is free CVAE-spec: all the mind GAN-CVAE: lover the mind has Music: mellow modular synth

Figure 4: Examples of generated lines for different instrumental music clips.

In total, 15 users participated in this study. As can be seen from Table 1, users preferred the lines from two experimental systems over the baseline regardless of the type of song played. The differences are statistically significant (ANOVA, $p<0.01$).

Study 2

For this study we asked five musicians to use the system as they played musical instruments. The participants were free to choose any musical instruments they liked and play any genre of music. Below is the summary of the instruments and genres played by the participants:

- Participant A: electric guitar / rock
- Participant B: keyboard / rock
- Participant C: guitar and electric drums / blues, rockabilly, punk, folk, rock.
- Participant D: piano / new age, contemporary classical, Bollywood, pop;
- Participant E: electric guitar / metal

The participants were asked to observe the generated lines as they played music, and answer the following questions after the completion of the study:

1. What were your thoughts after using the application?
2. What did you like about it?
3. What would you change about this system?
4. Would you consider using this system again?

Song description	**CVAE**	**CVAE-spec**	**GAN-CVAE**
Synthesizer, bass guitar, banjo, piano / forlorn, tense, slow	162	152	**195**
Electric guitar, synthesizer / harsh, aggressive, high-tempo	61	**94**	89
Modular synthesizer, keyboard / melancholy, low-tempo	64	48	**76**
Piano / sombre, tense, low-tempo	43	**91**	66
Modular synthesizer, keyboard / mellow, uplifting, ambient	53	72	53
Total	383	457	**479**

Table 1: Number of lines per system and instrumental song selected by the users in Study 1

5. Did the lines reflect the mood that your music conveys to you?

Since this is an exploratory study aimed at forming a better understanding of how artists may use such a system, we chose open-ended questions in order not to bias the participants and let them speak about their unique experiences while using the system and ways in which this interaction may have affected their creative processes.

All users found the experience enjoyable and would use the system again. Three users mentioned that they liked its simplicity and minimalist design. Interestingly, while our initial goal was to design a system to assist musicians in lyric writing, two unexpected new uses emerged from this study: (a) the system encouraged improvisation and trying new sounds and (b) the system was perceived as a helpful jam partner. This suggests that the system could be useful not only for lyric composition, but also for music playing in general. Below, we elaborate in more detail on the themes that emerged from the feedback.

Improvisation and experimenting with new sounds. Users commented that the system encouraged them to try new sounds and experiment with new musical expressions. User A: "[The system] inspires me to try things out to see what this sound does to the machine". User C: "there were a few times where the system started suggesting lyrics that got me thinking about structuring chords a bit differently and taking a song in different directions than originally intended".

User A also commented that the system encouraged them to be more experimental in their music: "You become much less critical of making unmusical sounds and you can explore the musical palette of the instrument to see what lines it provokes." This user also observed a game-like element of the system: "I am trying to make it say something crazy by playing crazy sounds" and "I am playing my guitar out of tune and try to find music in it as the machine is trying to find meaning in it." The user also noted that the generated lines sometimes encouraged them to stay in a certain music mode, such as minor chords, when they liked the lines that it generated. This suggests that the system could act as a responsive listener for musicians, to help them bring their music to the desired emotional expression. For example, if the musician intends their music to be sombre, and the emotions of the lines match the mood, they would know they are on track, and if the emotions in the lines start shifting as they try new expressions, it could be a signal that their music may not have the intended emotional effect.

System as a jam partner. One user commented that the system acts like a jamming partner as they play: "I don't feel like I am alone. I feel like jamming with someone." "It's like having an uncritical jam partner. It always responds to whatever you are trying." "The machine is trying to work with you even if you make a mistake." This suggests that a musician can benefit from using the system and be encouraged to play their instrument, even if they are not actively trying to write lyrics.

System as a source of lyrics. Users commented that the system would be useful as a source of new lyrical ideas. User C: "I could see it being very useful if I'm having a bit of writer's block and just noodling around to get a starting point." User B: "I like to jam with my band. We jam for 30 minutes or so. It would be nice to have it running. It would do the work for you as you are jamming. If at the end of it you have a pile of lyrics that would be great. That could be the starting material we could use for a song." The user emphasized that for them it would be important to see the lines after the jam session as they may miss some lines while playing their instruments: "I look at my hands when playing, occasionally glancing at the screen to see the lyrics. I would like to see the lines after the jam session."

Generated lines and music mood. Users noticed differences in lines, in particular with the changes in tempo and instruments used. One user played on their keyboard in the piano and Hammond B3 (organ) modes. The user played minor piano chords on the piano, noticing that the lines were darker in theme. Switching to organ with a more upbeat percussive sound and using major chords more often led to more upbeat lines.

Another user noticed that the system generated interesting lines when they played unusual musical phrases: "The system seemed to produce more interesting lines when it was fed slightly more novel audio, e.g. a tune that had a mode change (switching from major to minor)".

The genre of music did not appear to have as strong effect as tempo or the choice of instruments. One user who played blues, rock, folk and rockabilly genres did not observe a strong effect of genre. The likely reason is that the training dataset only contains rock music, and therefore the model has never seen music of other genres. It would be interesting to see how the lines are affected by genre, if the model is trained on a larger dataset, which is more representative of different genres.

Themes in generated lyrics Users noted presence of recurring themes as they played. They commented on often seeing recurring words in different lines shown to them when they played a musical composition. This is somewhat expected, since most musical compositions have repeating musical phrases which prompt the system to generate lines with similar words. Also, the frequently observed words noted by the users appeared to be different for each user, which suggests that the system differentiated between their musical styles, and narrowed down on specific themes. One user commented that when they experimented by playing songs with overtly obvious emotional valence (sad and happy), the system generated lines with markedly different words: "pain", "falling", "leaving" for a sad song, and "time", "dreams", "believe" for a happy song.

Training the model on a larger dataset would likely lead to more lexical diversity. Furthermore, during the experiments, the system always showed the users top two generated lines ranked by BERT, which was fine-tuned on a small dataset as well. When we changed this setting to sampling lines from the top 10 ranked lines, the diversity improved.

Related Work

For decades, music artists and poets used various techniques to find novel forms of lyrical and poetic expressions. The Dadaist movement of the early twentieth century gave rise to the cut-up technique, as a way to use randomness to free up the unconscious from the confines of logic and tradition (Lewis 2007). The technique was popularized by William Burroughs, who viewed language ("the Word") as a lock that restricts our creativity and confines us into predictable patterns of perception and expression. He saw cut-up as a way for both the artist and the reader to free themselves from this lock. Music artists, such as David Bowie and Kurt Cobain (Jones 2015), used cut-up technique extensively in their work to inspire creativity. In 1995, David Bowie and Ty Roberts co-created Verbasizer, a system that randomizes several text inputs and produces unexpected word combinations, which David Bowie then used as inspiration for his lyrics (Braga 2016). There has also been an interest in music and research communities to design systems for live musical improvisations. Notably, Biles (1994) designed GenJam, a system that generates jazz music improvisations. It was one of the pioneering works that envisioned a system as a co-creative partner in live music performance.

Automated creative text generation has long been getting attention from researchers at the intersection of computational creativity and artificial intelligence. Creative text generation includes tasks such as poetry (Manurung 2004; Zhang and Lapata 2014; Yang et al. 2017) and story (Riedl and Young 2010; Roemmele 2016; Brahman, Petrusca, and Chaturvedi 2020) generation. Recently automated lyric generation has also begun receiving attention from the research community.

Malmi et al. (2016) approached lyric suggestions from an information retrieval perspective. They selected the suitable next line from a dataset containing rap lyrics to produce a meaningful and interesting storyline. Similarly, Yu et al. (2019) developed a technique for cross-modal learning and used it to retrieve lyrics conditioned on the given audio. These systems, however, do not generate new lines, but instead provide the user with existing lines written by other artists.

With the recent advances in end-to-end text generation, lyrics generation is often considered a conditional text generation task. Tikhonov and Yamshchikov (2018) generated author-stylized poetry and lyrics using a language model conditioned on author embeddings. A system by Vechtomova et al. (2018) learned artist embeddings from the audio of their songs, and used them to condition the variational autoencoder in order to generate artist-stylized lyrics. Nikolov et al. (2020) generated rap verses conditioned on the content on any given text. Savery, Zahray, and Weinberg (2020) created a system for a real-time human and robot rap battle. They generated lyrics conditioned on the keywords present in the lyrics of the human rapper. Ackerman and Loker (2017) proposed a melody generation system, which was later extended to include generation of lyrics conditioned on the user-provided lyric lines and topic keywords. Cheatley et al. (2020) described a therapeutic use of this system.

While the above works generate lyrics conditionally, they do not take into account the characteristics of the music segment for which lyrics are generated. Watanabe et al. (2018) use a language model conditioned on the music score for lyrics generation. In our earlier work (Vechtomova, Sahu, and Kumar 2020) we developed a system to generate lyric lines conditioned on music audio. We showed that the system generated lyric lines that are consistent with the emotions evoked by a given music audio clip. In this work, we propose new approaches to align the learned latent spaces of audio and text representations, and implement real-time lyric generation for live music.

Implementation Details

Text-CVAE. We use the Tensorflow framework (Abadi et al. 2016) to implement the text-CVAE. The encoder is a single-layer bi-LSTM and a decoder is an LSTM. The hidden state dimensions were set to 300 and the latent space to 128. The CVAE models in our experiments were trained for 500 epochs.

Spec-VAE. PyTorch (Paszke et al. 2019) is used to implement the spec-VAE. The encoder has four Conv2d layers interleaved with ReLU activation function (Nair and Hinton 2010). The decoder is a mirror image of the encoder, with four ConvTranspose2d layers, interleaved with three ReLU activations and one Sigmoid activation in the end. We use 128 latent dimensions for the mean and sigma vectors. During training, we use a batch size of 32, learning rate of 1e-4, and Adam optimizer (Kingma and Ba 2015). The sampling temperature is 1.0 for both training and inference.

GAN. We use the AllenNLP library (Gardner et al. 2018) to implement the GAN. The generator and discriminator networks are 3-layered feed-forward neural networks, interleaved with ReLU activation function. During training, we use a batch size of 32, learning rate of 1e-3, and Adam optimizer for both the generator and the discriminator. We set

$\lambda_{MSE} = 1.0$ to ensure diverse lyric lines. The sampling temperature is 1.0 during both training and inference. The GAN alignment network was trained for six epochs.

BERT. We use the Transformers library (Wolf et al. 2020) to fine-tune a BERT-base model for sequence classification on our custom dataset. The model is trained for 15 epochs using a learning rate warmup scheduler for the first 500 steps of training with a weight decay of 0.01. We use a batch size of 16 for training and 64 for inference.

Conclusions

We developed a system for real-time generation of lyrics matching the live instrumental music being played by an artist during a jam session. Two novel approaches to align the learned representations of music and lyrics have been proposed: GAN-CVAE, which adversarially learns to predict the lyric representation from the music representation, and CVAE-spec, which transfers the topology of the music (spectrogram) latent space learned by the spectrogram VAE to the lyric latent space learned by the text CVAE. Our user study showed that users selected the lines generated by the proposed two methods significantly more often than the lines generated by a baseline for different types of instrumental songs. For another user study we recruited musicians performing live music. Their feedback suggested that the system could be useful not only for lyric writing, but also (a) to encourage musicians to improvise and try new musical expressions, and (b) act as a non-critical jam partner.

The user studies in this work as well as statistical analysis in our prior research indicate that it is possible to learn semantic and emotional associations between music audio and lyric texts in an unsupervised manner. As the follow-up to this work, we are conducting further analysis of what other associations between lyric texts and music the unsupervised neural network models are learning.

Acknowledgments

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant No. RGPIN-2019-04897 and Compute Canada (www.computecanada.ca).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Ackerman, M., and Loker, D. 2017. Algorithmic songwriting with ALYSIA. In *International conference on evolutionary and biologically inspired music and art*, 1–16. Springer.

Biles, J. 1994. Genjam: A genetic algorithm for generating jazz solos. In *ICMC*.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Józefowicz, R.; and Bengio, S. 2016. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 10–21.

Braga, M. 2016. The Verbasizer was David Bowie's 1995 lyric-writing music app.

Brahman, F.; Petrusca, A.; and Chaturvedi, S. 2020. Cue me in: Content-inducing approaches to interactive story generation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 588–597. Suzhou, China: Association for Computational Linguistics.

Cheatley, L.; Ackerman, M.; Pease, A.; and Moncur, W. 2020. Co-creative songwriting for bereavement support. In *Eleventh International Conference on Computational Creativity: ICCC'20*, 33–41. Association for Computational Creativity.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N.; Peters, M.; Schmitz, M.; and Zettlemoyer, L. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Jones, J. 2015. How David Bowie, Kurt Cobain, Thom Yorke write songs with William Burroughs' cut-up technique.

Khan, K.; Sahu, G.; Balasubramanian, V.; Mou, L.; and Vechtomova, O. 2020. Adversarial learning on the latent space for diverse dialog generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, 5026–5034. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In Bengio, Y., and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P., and Welling, M. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kolchinsky, A.; Dhande, N.; Park, K.; and Ahn, Y.-Y. 2017. The minor fall, the major lift: inferring emotional valence of musical chords through lyrics. *Royal Society open science* 4(11):170952.

Lewis, P. 2007. *The Cambridge Introduction to Modernism*. Cambridge Introductions to Literature. Cambridge University Press.

Malmi, E.; Takala, P.; Toivonen, H.; Raiko, T.; and Gionis, A. 2016. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 195–204.

Manurung, H. 2004. An evolutionary algorithm approach to poetry generation.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Icml*.

Nikolov, N. I.; Malmi, E.; Northcutt, C.; and Parisi, L. 2020. Rapformer: Conditional rap lyrics generation with denoising autoencoders. In *Proceedings of the 13th International Conference on Natural Language Generation*, 360–373.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. 8024–8035.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Roemmele, M. 2016. Writing stories with help from recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Savery, R.; Zahray, L.; and Weinberg, G. 2020. Shimon the rapper: A real-time system for human-robot interactive rap battles. In *Eleventh International Conference on Computational Creativity: ICCC'20*.

Tikhonov, A., and Yamshchikov, I. P. 2018. Guess who? multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, 787–794.

Vechtomova, O.; Bahuleyan, H.; Ghabussi, A.; and John, V. 2018. Generating lyrics with variational autoencoder and multi-modal artist embeddings. *arXiv preprint arXiv:1812.08318*.

Vechtomova, O.; Sahu, G.; and Kumar, D. 2020. Generation of lyrics lines conditioned on music audio clips. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, 33–37. Online: Association for Computational Linguistics.

Watanabe, K.; Matsubayashi, Y.; Fukayama, S.; Goto, M.; Inui, K.; and Nakano, T. 2018. A melody-conditioned lyrics language model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 163–172.

Willimek, D., and Willimek, B. 2014. Why do minor chords sound sad? the theory of musical equilibration and the emotions of chords. *Journal of Psychology & Psychotherapy* 4:139.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Yang, X.; Lin, X.; Suo, S.; and Li, M. 2017. Generating thematic chinese poetry using conditional variational autoencoders with hybrid decoders. *In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.*

Yu, Y.; Tang, S.; Raposo, F.; and Chen, L. 2019. Deep cross-modal correlation learning for audio and lyrics in music retrieval. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 15(1):1–16.

Zhang, X., and Lapata, M. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 670–680.