Multimodal Transformer-Based Lyric Generation from MIDI and Text Data

Stanford CS224N Custom Project

Vivek Vajipey
Department of Computer Science
Stanford University
vvajipey@stanford.edu

Anthony Zhan
Department of Computer Science
Stanford University
azhan9@stanford.edu

Steven Zhao
Department of Computer Science
Stanford University
szhao415@stanford.edu

Abstract

Current generative language models are designed to take a text prompt as input, and output the highest probability continuation. However, when applying this idea to the design space of lyric generation, there is an obvious shortcoming — lyrics are often created with music's auditory features in mind, such as tempo, tonality, melody, rhythm, etc. Which is to say, a model able to interpret both text prompts and snippets of music may be able to produce much better music. Therefore, our project seeks to combine both text and music data in a multimodal transformer-based model in order to generate lyrics more effectively than a purely text-based model, utilizing multiple transformer architectures, including MusicBERT and GPT-2.

1 Introduction

For any cohesive song, lyrical content is always inseparable from the musical content. Lyrics must fit the melody, rhythm, harmonies, etc. for the song to sound good, and songwriters typically try to match lyrics with the music as best as possible. However, currently-existing methods of natural lyric generation typically generate from purely text inputs [1] [2]. The few that use information other than text tend to use broad categories such as positive and negative sentiment or music style that don't provide detailed enough information about the desired lyrics [3] [4] . We propose creating a multimodal, transformer-based model that generates lyrics based on a text prompt and MIDI file. More specifically, we create a model that uses MusicBERT to encode data from the MIDI into embeddings, projects these embeddings into the text embeddings space, and feed the projected music and text embeddings together into a finetuned GPT2 to generate text. Overall, the approach we use in this project is a novel use of transformer based models that brings in multimodal inputs to augment the types of generated outputs.

2 Related Work

Much of previous work has focused on creating a natural lyric generator based on only text inputs. Gill et al. created an LSTM model that produces lyrics for specific genres, and claim that the model works well in pop and rap [2]. Chuang et al. made a GAN for Chinese rap lyrics titled RapGAN that outperformed other lyric generators on human evaluation. Some have also tried transformers and achieved strong results [1]. Overall however, much of the literature on lyric generation uses older models (e.g. LSTMs) that typically don't perform as well as transformers. Thus, our first insight is that we would like to incorporate transformers in our own model, and thus contribute to the literature on transformers in lyric generation.

Some lyrics generators have tried to incorporate information outside of text. Bao et al. created a model that takes into account positive or negative sentiment of the song [3]. However, positive and negative are very broad terms, and only provide limited information about how the lyrics should be constructed. Chang et al. created a transformer-based model that generates lyrics in accordance to specific genres such as "pop" and "rock" [4]. While this perhaps is more specific than sentiment, such genre labels are also quite broad. For example, punk rock and psychedelic rock are extremely different lyrically, yet they would both be categorized under "rock". Therefore, our second insight is that we want to provide a non-text context that is more nuanced and detailed than what people have tried before.

There has been existing work in using transformer-based models to understand music in the same way as language. MusicBERT, which we explain in the next section, is a RoBERTa-based model trained on a dataset of over 1 million MIDI songs for several downstream tasks such as genre classification and melody prediction [5].

3 Approach

We propose MLG (MIDI-conditioned Lyric Generation), a multimodal, transformer-based model that generates lyrics based on combined MIDI and text data. The architecture is based on GPT-2, a decoder-based language model provided by OpenAI [6], but with several key changes. Specifically, we use the Huggingface transformers GPT-2 model[1] with 124M parameters, which we modify to accept embeddings corresponding to MIDI data (see Figure 1).

MIDI files — symbolic representations of music, akin to sheet music — have several advantages over raw audio files. Most importantly, MIDI files are much more *compact*, which means building and processing the dataset is more efficient, and MIDI data is *discretized* rather than continuous. In order to convert the MIDI data into a form that GPT-2 can recognize, we perform the following preprocessing:

- A MIDI file is converted into tokens using the Octuple tokenizer.
- The first $n = 512$ tokens are fed into the MusicBert model, a large pre-trained model for symbolic music understanding [5].
- The outputted hidden states (of size $h_{\text{MusicBERT}} = 768$) are saved as a Pytorch tensor of shape $n \times h_{\text{MusicBERT}}$.

The goal is to use MusicBERT, which has been pre-trained on MIDI data, to extract some high-level meaning from the MIDIs, instead of learning the MIDI embeddings from scratch. Our motivation

[1] https://huggingface.co/gpt2

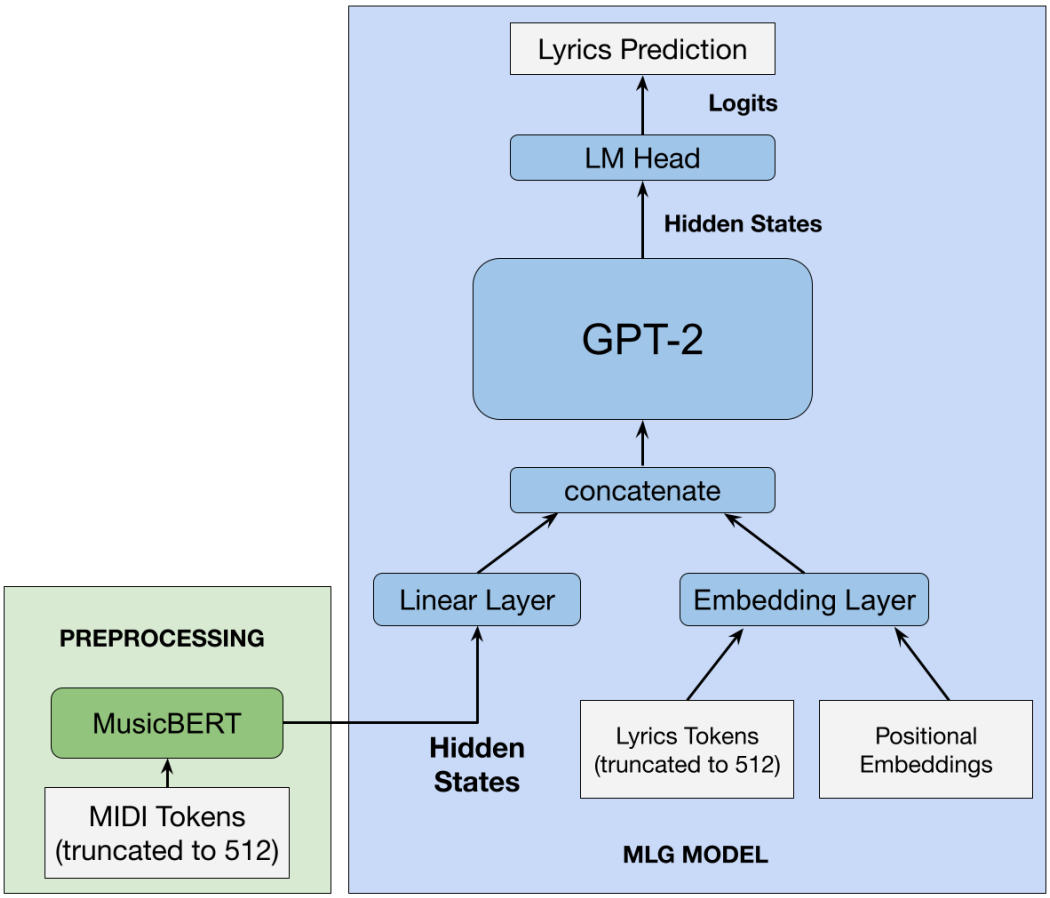

Figure 1: Proposed model architecture

for this step is that learning embeddings from scratch would require resources that we didn't have; in addition, our dataset is too constrained to produce embeddings as expressive and powerful as MusicBERT.

Before inputting into GPT-2, we incorporate a linear layer with weight matrix and bias W and b, of dimensionality $h_{\text{MusicBERT}} \times h_{\text{GPT-2}}$ and $h_{\text{GPT-2}}$ respectively, where $h_{\text{MusicBERT}} = 768$ is the hidden size for MusicBERT and $h_{\text{GPT-2}} = 768$ is the hidden size for GPT-2. This layer essentially learns how to best project vectors from the MusicBERT embedding space to the GPT-2 embedding space.

The projected embeddings are concatenated with the text embeddings (obtained by adding token embeddings and position embeddings), and the entire sequence of embeddings is fed into GPT-2. Note that since GPT-2 accepts token IDs as input by default, we had to modify the `forward()` function to accept input embeddings instead.

Besides the embedding layer, the rest of the GPT-2 architecture (i.e. stacked decoders) is unchanged, and the final hidden states are passed through an LM head to obtain the logits.

Note that even though the output consists of music and text data, the model output, i.e. the generated lyrics, is strictly text. For training, the loss is computed only between the predicted lyrics and reference lyrics. Thus the model learns to generate lyrics *conditioned on* the background music.

To evaluate performance, we compare our models to a GPT-2 model finetuned on lyrics without MIDI data. By comparing this ablation with the multimodal model, the intention is to establish if providing MIDI data truly helps to generate better lyrics.

4 Experiments

For the MIDI data, we use the current largest open-source collection of MIDI files, the Lakh MIDI Dataset compiled by Raffel [7]. It consists of 176,000 MIDI files scraped from online sources. Although the Lakh MIDI Dataset is large in size, most of the metadata and formatting is disorganized and unstandardized. Moreover, a large percentage of the songs with lyrics are in languages other than English, such as German, Vietnamese, and Portugeuse. After filtering the dataset for MIDI files with lyrics in the metadata and further selecting English songs, only around 3,000 files remained. Thus, it was necessary to augment the dataset by finding the lyrics associated with the Lakh MIDI Dataset songs. To aid in this effort, we used the Lakh MIDI Dataset Clean[8], which aligns a 45,000 MIDI file subset of the original Lakh MIDI Dataset with the artists and song titles of the songs.

With the Lakh MIDI Dataset Clean's reliable labels for the artist names and song titles of the MIDI files, we created a Python script to retrieve the corresponding lyrics using the Genius API [9] and the lyricsgenius library [10]. The final dataset we used contains approximately 13,600 songs, which we split into train, validation, and test sets with an 80-10-10 split.

In terms of evaluation methods, we will be relying on qualitative evaluation of the generated lyrics for our model. This is because traditional quantitative metrics for natural language generation, such as BLEU or perplexity, are distinct from lyrics quality. For example, BLEU uses bag-of-words to measure the similarity between two sentences. The model may output lyrics that are as good as the original lyrics but are significantly different; BLEU would be unable to capture this. Thus, we will primarily be qualitatively evaluating MLG on the following two criterion: first, how coherent are the lyrics? Are they grammatically correct, do they respect song structure, and are they spelled correctly? Second, are the lyrics suitably original? While we expect some repetition due to the inherent repetitiveness of most song lyrics, we hope that MLG will be capable of creating some original lyrics.

For our results, we noticed that the lyrics outputted were practically always repetitive, usually repeating verse after verse. For example, we observed the following output after inputting "Stop" by Spice Girls:

Comparison

Input:
Spice_Girls Stop Lyrics
Verse 1: Ginger, Sporty, Baby & Posh

You just walk in, I make you smile
It's cool but you don't even know me
You take an inch, I run a mile
Can't win you're always right behind me
And we know that you could go and find some other
Take or leave it or just don't even bother
Ca

Model Output:
using me to lose my mind

[Verse 2: Ginger, Sporty, Baby & Posh]
You just walk in, I make you smile
It's cool but you don't even know me
You take an inch, I run a mile
Can't win you're always right behind me
And we know that you could go and find some other
Take or leave it or just don't even bother
Causing me to lose my mind

[Verse 3: Ginger, Sporty, Baby & Posh] You just walk in, I make you smile It's cool but you don't even know me You take an inch, I run a mile Can't win you're always right behind me And we know that you could go and find some other Take or leave it or just don't even bother Causing me to lose my mind

As we can see from above, the model tends to repeat the same verse multiple times; this occurred despite us adjusting the model's temperature several times. Obviously then, the model is generally grammatically correct, as it usually repeats verses. More significantly however, the model does appear to understand what the structure of lyrics are. For example, it labels one stanza "Verse 2", and then the next stanza "Verse 3". Other examples also have numerically correct verses. Thus, this may not simply be an issue of an NLP model repeating itself numerous times; otherwise, it would not understand what number should be written for each verse.

Obviously, the model does not do as well on originality as we initially hoped. The model's output is more repetitive than we would've liked. However, we do believe that the model is capable of generating some coherent lyrics. For example, it generates "Causing me to lose my mind" on its own, despite that not being in the input.

5 Analysis

Overall, the model did not perform as well as we hoped. We attribute this to two primary limitations. First, data quality was lacking. We had to scrape for the data on our own, and we were not able to come up with many training examples. Moreover, the lyrics come from Genius, and are thus largely mainstream pop and other repetitive genres. Because of how repetitive pop lyrics are, it is rather unsurprising that a model trained on pop lyrics would also be repetitive. We would have found a more varied dataset if one existed.

Additionally, another issue is that GPT-2 can only take in at most 1024 tokens. This means that first, when we were training, the length of the training examples we could use were constrained. Additionally, the number of MIDI tokens we could incorporate were also limited to 512, since we split the space for MIDI and GPT-2 equally. However, a full song has many more than 512 tokens; the

small amount of tokens GPT-2 can take in prevented us from obtaining more complete representations of songs to feed into the model.

6 Conclusion

In conclusion, we were able to create a new, unique architecture that takes in both music embeddings and text to generate lyrics. However, due to constraints from the quality and quantity of our data, and the limited size of tokens that GPT-2 can take in, our model ended up much more repetitive than we hoped for. We hope that in the future, others will be able to make improvements on the basis of the original architecture that we created.

We would like to acknowledge John Thickstun (Postdoctoral scholar, Stanford University Department of Computer Science) for his consultation in building the MIDI preprocessing step and designing of model architecture.

References

[1] Evan Crothers, Herna Viktor, and Nathalie Japkowicz. In bloom: Creativity and affinity in artificial lyrics and art, 2023.

[2] Harrison Gill, Daniel Lee, and Nick Marwell. Deep learning in musical lyric generation: An lstm-based approach. *The Yale Undergraduate Research Journal*, 1(1), 2020.

[3] Chunhui Bao and Qianru Sun. Generating music with emotions. *IEEE Transactions on Multimedia*, pages 1–1, 2022.

[4] Jia-Wei Chang, Jason C. Hung, and Kuan-Cheng Lin. Singability-enhanced lyric generator with music style transfer. *Computer Communications*, 168:33–53, 2021.

[5] Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu. Musicbert: Symbolic music understanding with large-scale pre-training, 2021.

[6] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[7] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

[8] Sparsh Gupta. Lakh midi clean. `https://www.kaggle.com/datasets/imsparsh/lakh-midi-clean`, 2018.

[9] Genius. Genius api documentation, 2023.

[10] XYZ Corporation. Api documentation for xyz. Documentation, XYZ Corporation, 2022. Accessed on March 5, 2023.