ECE 1786 - Creative Applications of Natural Language Processing

ArtIstic GENREator: Final Report

December 13, 2022

Team:

Yiqian Qin (1007014507) | ECE MEng

Wenzhe Xu (1001192666) | MScAC

Word Count: 1999

0. Permissions

Yiqian and Wenzhe both agree on the following schemes:

- permission to post video: wait till see video
- permission to post final report: yes
- permission to post source code: no

1. Introduction

Natural language models and techniques have developed substantially and demonstrated great promise in automatically producing written or spoken narratives by extracting information from a large corpus. However, some creative writing tasks, such as lyric writing, require not only creativity but also domain skills to produce complex patterns with unique aesthetic qualities, such as flow, rhyme, and repetition. Moreover, musical genres are remarkably different in their lyrical styles, which is reflected in linguistic features, including line length, word variation, and themes.

With that in mind, the goal of our project is to explore the capabilities of transformer-based language models to generate lyrics. Specifically, we focus on lyric generation given a specific musical genre and user-specified starting prompt, and aim to generate lyrics that preserve features native to the specified genre, but not identical to the existing lyrics.

2. Illustration / Figure

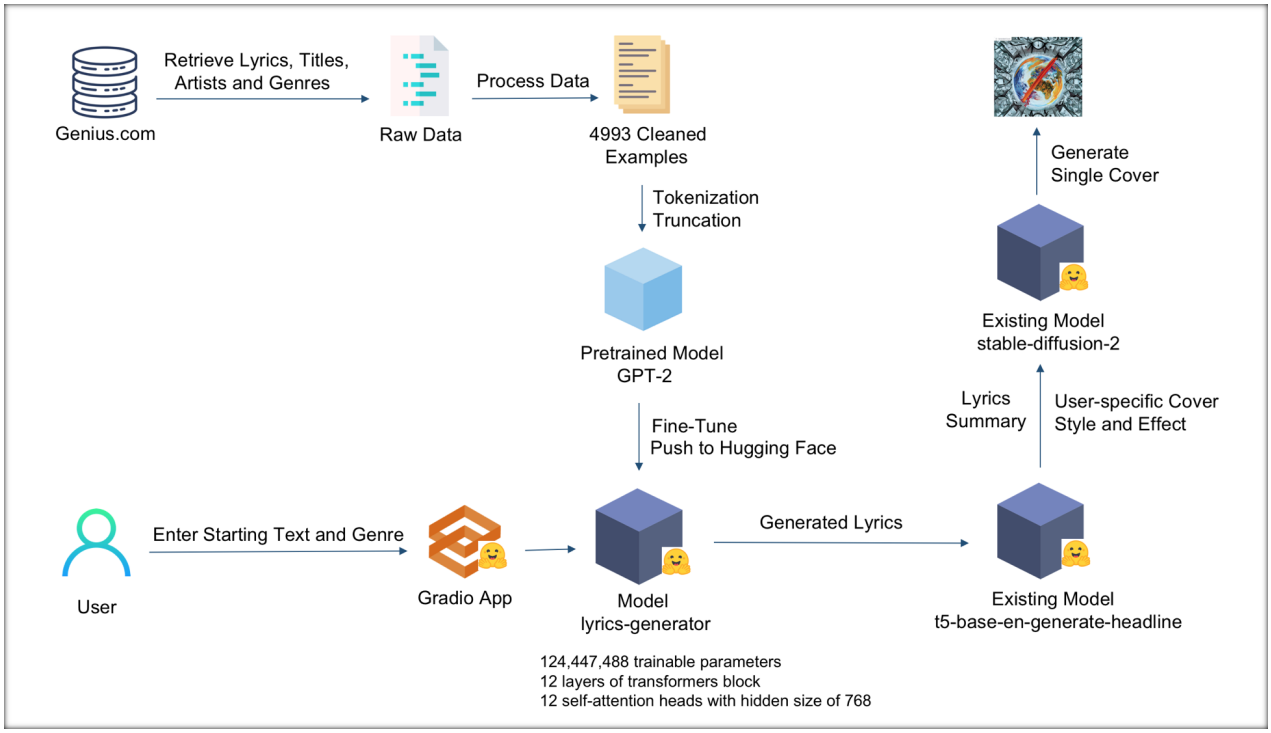


Fig. 1: Core idea and architecture of the project

3. Background & Related Work

Recent work [1] has shown the effectiveness of transformer-based language models for text generation. The authors adopted multi-task learning with GPT-2 to generate reasonable stories with logic and global coherence from a leading context. The results are inspiring, as the model was able to learn various causal and temporal dependencies between the sentences in the stories.

Text generation for artistic purposes such as poetry and lyric writing has also been explored in recent years [2]. In their works, the authors employed a LSTM network to produce lyrics for a specific genre given an input sample lyric, and explored two transformer-based models, BERT and GPT-2, to experiment with lyric generation. They obtained only mediocre results with GPT-2, partly due to GPT-2 being pre-trained on a large corpus of prose-like text, which is fairly different from how lyrical text is structured.

As one of the major foreseeable challenges of adapting a large, pre-trained model to a specific task is to break certain ingrained habits of the large general model, we deploy and examine different data-processing and training strategies as an attempt to improve the model performance.

4. Data and Data Processing

We fetched song lyrics and other relevant information from Genius.com, an online music and lyrics database. Specifically, we wrote a *retrieve_song* function with the LyricsGenius library to retrieve track titles, lyrics and artists based on specific genres. We focused on 5 genres (Pop, Rap, Country, Rock and R&B), and collected 999, 1000, 994, 1000, 1000 training examples respectively.

To process raw lyrics data for each track before training and fine-tuning, we created a *preprocess_lyrics* function that removes the irrelevant information from the lyrics: the track title and language, the "Embed" information at the end of the lyrics, all square brackets that describe the song structure, the advertisement "You might also like", the information on live tickets, the leading and trailing whitespace characters, and the extra newline characters between verses. The function then adds the tokens that indicate the beginning and the end of the lyrics for model training purposes. Additionally, in order to specify the genre as an input to the model, a special genre-specific token (e.g. <pop>) is added at the beginning of the lyrics to indicate its genre.

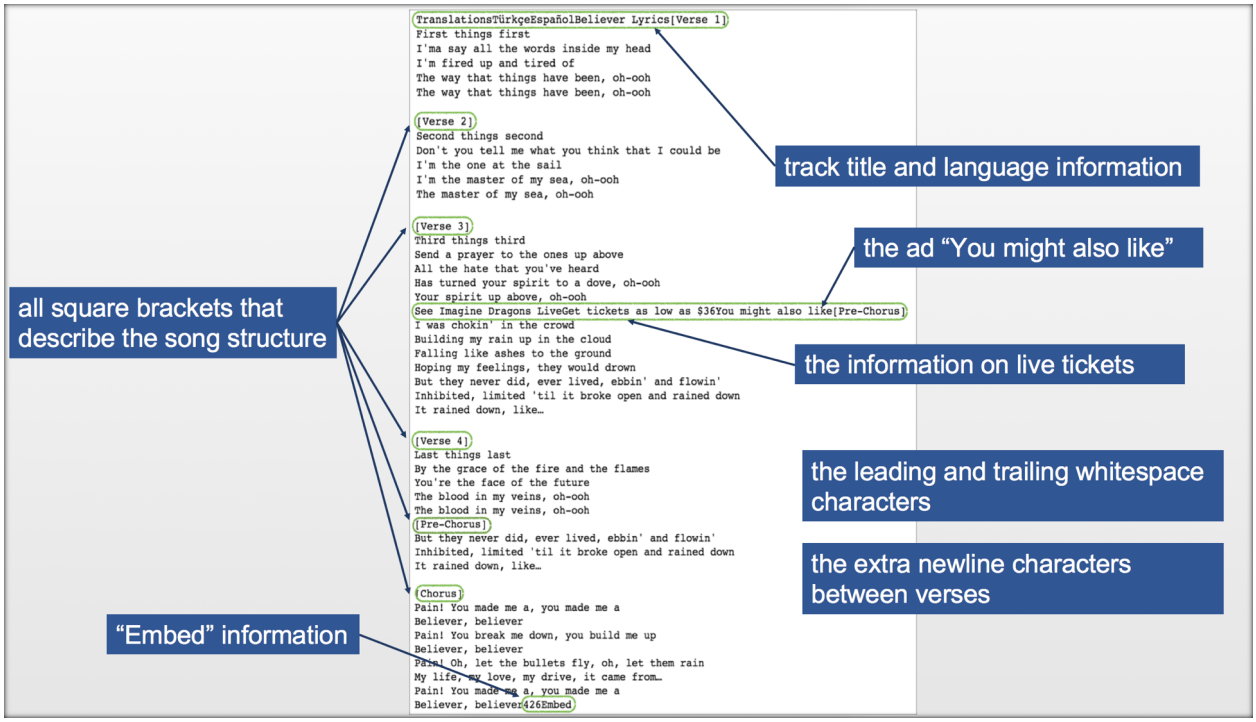


Fig. 2: Sample lyrics before data processing

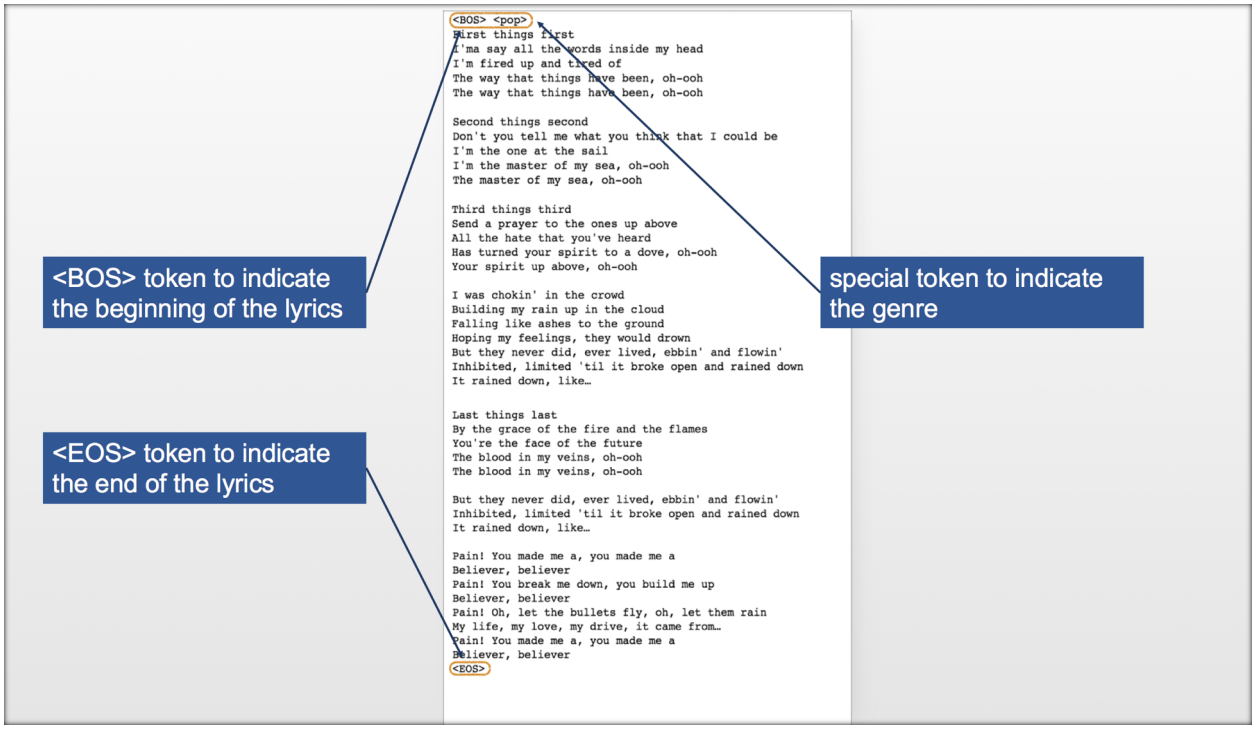


Fig. 3: Sample lyrics after data processing

Title	Artist	Lyrics	Genre
The Sound of Silence	['Simon', 'Garfunkel']	<BOS> <pop>\nHello darkness, my old friend\nI've come to talk with you again\nBecause a vision softly creeping\nLeft its seeds while I was sleeping\nAnd the vision that was planted in my brain\nStill remains within the sound of silence\n\nIn restless dreams, I walked alone\nNarrow streets of cobblestone\n'Neath the halo of a street lamp\nI turned my collar to the cold and damp\nWhen my eyes were stabbed by the flash of a neon light\nThat split the night, and touched the sound of silence\n\nAnd in the naked light I saw\nTen thousand people, maybe more\nPeople talking without speaking\nPeople hearing without listening\nPeople writing songs that voices never shared\nAnd no one dared disturb the sound of silence\n\n"Fools," said I, "You do not know\nSilence like a cancer grows\nHear my words that I might teach you\nTake my arms that I might reach you"\nBut my words, like silent raindrops, fell\nAnd echoed in the wells of silence\n\nAnd the people bowed and prayed\nTo the neon god they made\nAnd the sign flashed out its warning\nIn the words that it was forming\nAnd the sign said, "The words of the prophets are written on the subway walls and tenement halls\nAnd whispered in the sound of silence"\n<EOS>	Pop

Fig. 4: Sample training example after data processing

Genres	Cleaned Examples	Total Number of Cleaned Examples	Examples for Training	Examples for Validation
Pop	999	4993 (proven to be sufficient for this project)	4493 (90% of the total examples)	500
Rap	1000			
Country	994			
Rock	1000			
R&B	1000			

Table. 1: Summary statistics of training data

5. Architecture and Software

Our lyrics generation model is based on the standard OpenAI GPT-2 model, a large transformer-based language model that uses 12 layers of transformers block and 12 self-attention heads with a hidden size of 768. The pretrained GPT-2 model was instantiated with Hugging Face's generic model class *AutoModelForCausalLM*, which creates a causal language modeling head for text generation. The model had 124,439,808 parameters before fine-tuning, and the number of parameters increased to 124,447,488 after we introduced the additional special tokens related to music genres. Our lyrics generator was trained on the complete training set (4493 examples) for 10 epochs, and tested on 500 different songs. The training arguments for reproducing a model similar to ours are listed below.

We also extended the project by adding the functionality to generate the single cover based on the machine-generated lyrics. After generating the lyrics, a one-line summary is created with *t5-base-en-generate-headline*, a model based on the text-to-text transfer transformer (T5) and pre-trained on a collection of 500,000 articles. The lyrics summary is then combined with the user-specified cover style and effect to produce a text prompt, which serves as the input to the text-to-image model that generates the single cover artwork. After multiple rounds of testing, we chose *stable-diffusion-2* as our text-to-image model based on the quality of the generated images.

It is a diffusion-based model pre-trained on an aesthetic subset of LAION-5B, a CLIP-filtered dataset of 585 billion image-text pairs.

```
training_args = TrainingArguments(
    output_dir=output_dir,
    evaluation_strategy="steps",
    prediction_loss_only=False,
    logging_steps=100,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=10,
    save_steps=500,
    learning_rate=5e-5,
    weight_decay=0.01,
)
```

Fig. 5: Training arguments

6. Qualitative Results

As we opted not to build a baseline model, the qualitative comparison is among the generated lyrics across different user-specified music genres. We first wrote several different starting prompts. For each starting prompt, we then added the special token that represents one of the 5 genres to create 5 different prompts. These prompts were then fed into our lyrics generator. Some of the starting prompts and the generated lyrics are listed below.

The overall quality of the machine-generated lyrics was quite good. Most often the generated lyrics are meaningful and coherent, and they almost never get stuck to the starting prompt. The model successfully captured some distinct linguistic features of the training corpus of lyrics, such as the use of newline characters to separate different sections (verse / chorus / bridge) of the song, and the propensity for certain words and phrases to be repeated. Moreover, the generated lyrics are semantically apt in their relevant genre and often took on real lyrical flows and structures. Some examples include the tendency of boastful (Fig. 6) and sometimes insulting lyrics (Fig. 10) in rap music, the focuses on love, loneliness, and work in country music (Fig. 7), and the theme of rebellion and genre-specific word usage (e.g. “drum”, “roar”, “rock”) in rock music (Fig. 8, 11).

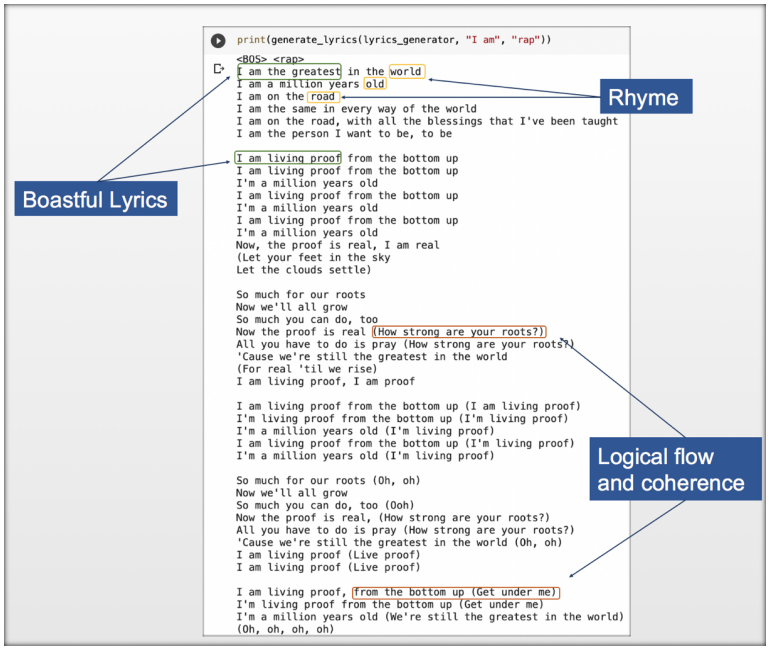


Fig. 6: Sample generated lyrics ("I am…", Rap)

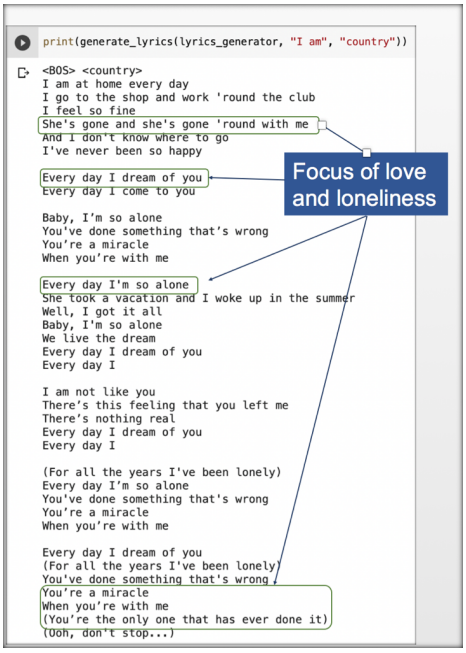


Fig. 7: Sample generated lyrics ("I am…", Country)

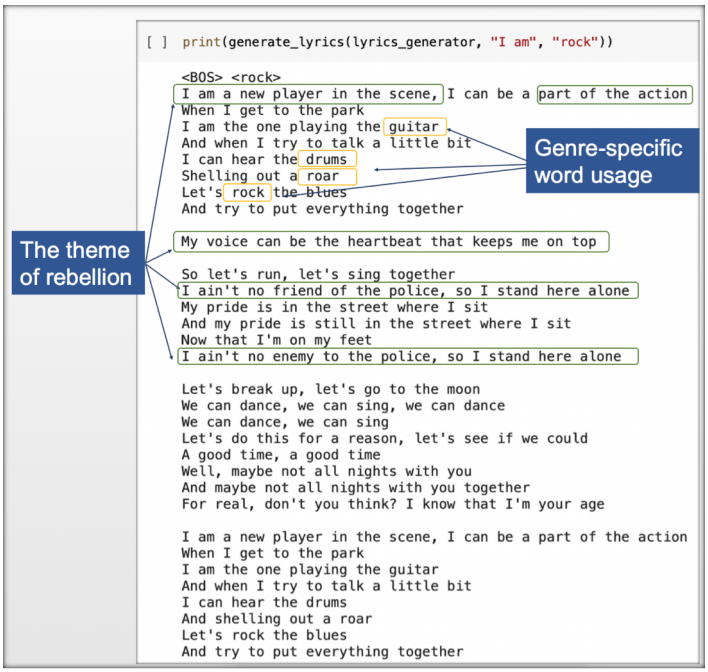


Fig. 8: Sample generated lyrics ("I am…", Rock)

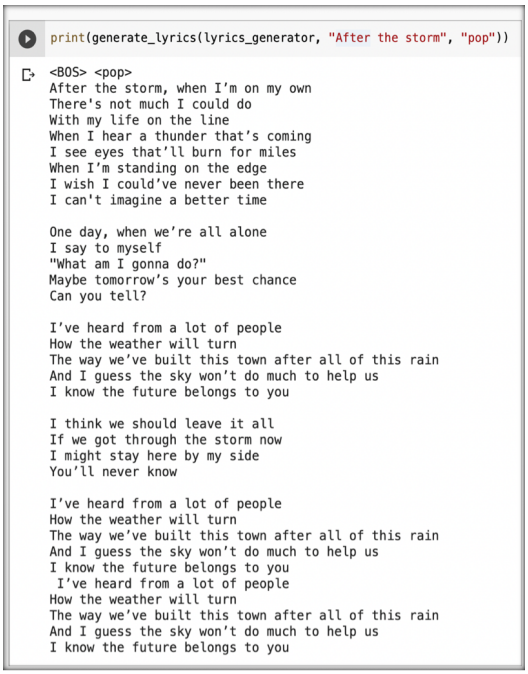
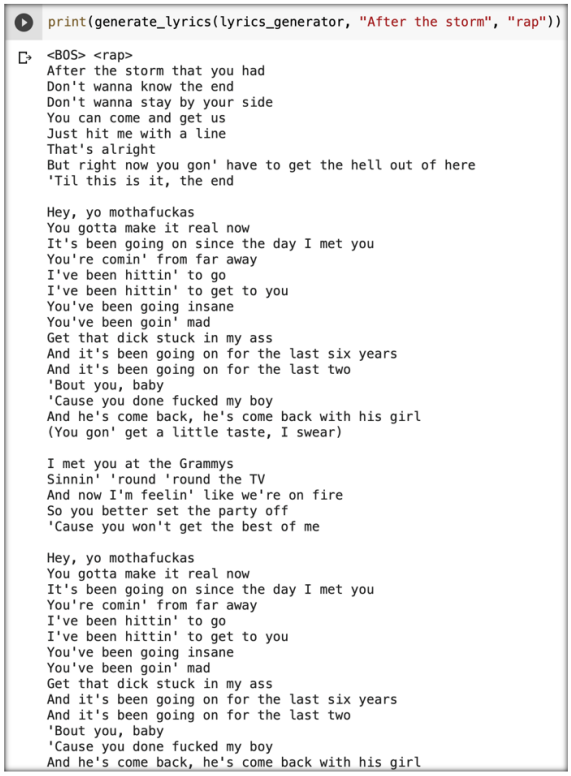


Fig. 9: Sample generated lyrics ("After the storm…", Pop)



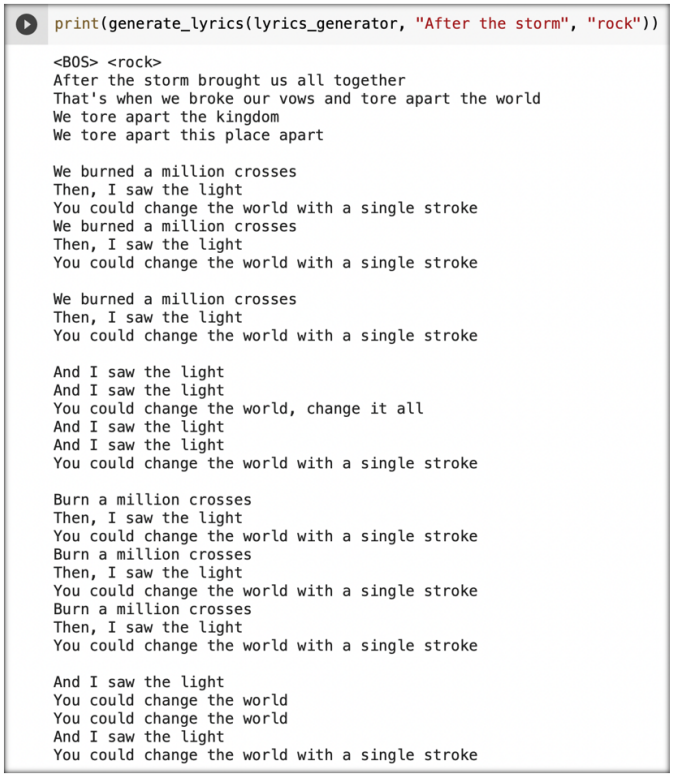
```
print(generate_lyrics(lyrics_generator, "After the storm", "rap"))

<BOS> <rap>
After the storm that you had
Don't wanna know the end
Don't wanna stay by your side
You can come and get us
Just hit me with a line
That's alright
But right now you gon' have to get the hell out of here
'Til this is it, the end

Hey, yo mothafuckas
You gotta make it real now
It's been going on since the day I met you
You're comin' from far away
I've been hittin' to go
I've been hittin' to get to you
You've been going insane
You've been goin' mad
Get that dick stuck in my ass
And it's been going on for the last six years
And it's been going on for the last two
'Bout you, baby
'Cause you done fucked my boy
And he's come back, he's come back with his girl
(You gon' get a little taste, I swear)

I met you at the Grammys
Sinnin' 'round 'round the TV
And now I'm feelin' like we're on fire
So you better set the party off
'Cause you won't get the best of me

Hey, yo mothafuckas
You gotta make it real now
It's been going on since the day I met you
You're comin' from far away
I've been hittin' to go
I've been hittin' to get to you
You've been going insane
You've been goin' mad
Get that dick stuck in my ass
And it's been going on for the last six years
And it's been going on for the last two
'Bout you, baby
'Cause you done fucked my boy
And he's come back, he's come back with his girl
```



```
print(generate_lyrics(lyrics_generator, "After the storm", "rock"))

<BOS> <rock>
After the storm brought us all together
That's when we broke our vows and tore apart the world
We tore apart the kingdom
We tore apart this place apart

We burned a million crosses
Then, I saw the light
You could change the world with a single stroke
We burned a million crosses
Then, I saw the light
You could change the world with a single stroke

We burned a million crosses
Then, I saw the light
You could change the world with a single stroke

And I saw the light
And I saw the light
You could change the world, change it all
And I saw the light
And I saw the light
You could change the world with a single stroke

Burn a million crosses
Then, I saw the light
You could change the world with a single stroke
Burn a million crosses
Then, I saw the light
You could change the world with a single stroke
Burn a million crosses
Then, I saw the light
You could change the world with a single stroke

And I saw the light
You could change the world
You could change the world
And I saw the light
You could change the world with a single stroke
```

Fig. 10: Sample generated lyrics ("After the storm…", Rap)

Fig. 11: Sample generated lyrics ("After the storm…", Rock)

7. Quantitative Results

One of the challenges for text generation tasks is finding the suitable performance metrics to quantitatively compare and evaluate the results. Standard metrics such as BLEU and ROUGE scores are designed to evaluate the quality of machine-translated text, and therefore not considered as the ideal choice for our lyrics generation model. Inspired by the ideas in [2], we devised and implemented four rudimentary metrics, namely average line length, word repetition, word variation, and point-of-view, to assess the similarities and differences between the corpus and the generated lyrics. The metrics are defined as follows.

- Average Line Length - average number of words in each line of a song
- Word Repetition - number of occurrences of repeated words in a song, normalized by the total number of words
    - All unigram, bigram, and trigram sequences are taken into account
    - e.g. "Oh I oh I oh I oh I, I'm in love with your body" - *word repetition* = 3
- Word Variation - number of unique words in a song, normalized by the total number of words
- Point-of-View - difference between the number of lines that start with "I" and the number lines that start with "You", normalized by the total number of lines

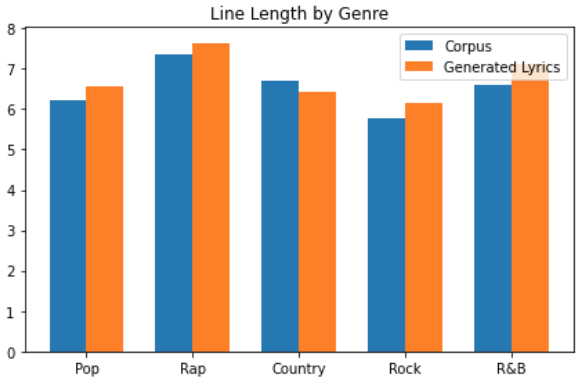


Fig. 12: Line length by genre

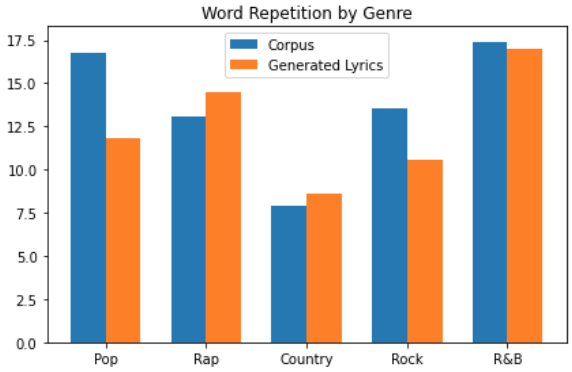


Fig. 13: Word Repetition by Genre

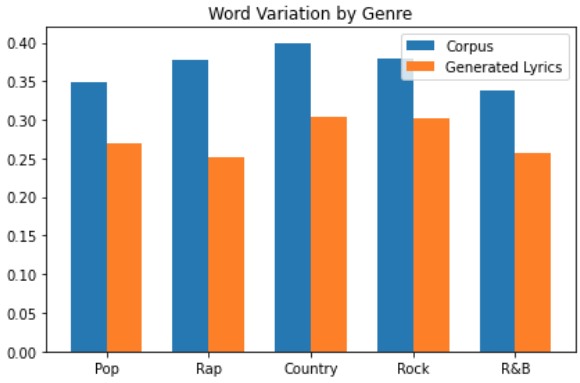


Fig. 14: Word Variation by genre

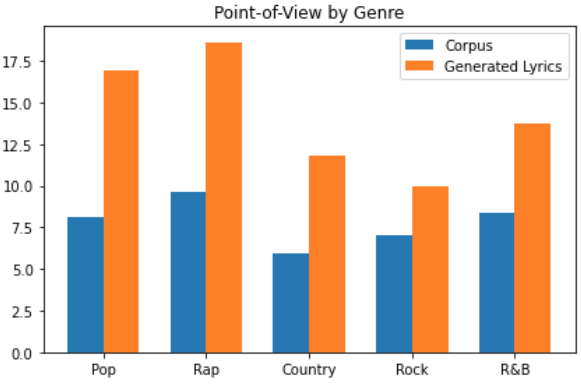


Fig. 15: Point-of-View by Genre

500 songs (100 per genre) were randomly generated with our model and evaluated on the four self-devised metrics. The comparison between the corpus and the generated lyrics across each of the five genres are shown in Fig. 12-15. When evaluated on these metrics, the generated lyrics are expected to follow a similar trend as the corpus used to train the model. For example, according to Fig. 13, country music has a much lower number of word repetitions compared to other genres, and we indeed observe similar results in the generated lyrics. The Pearson correlation coefficient is used as the final scalar output to quantitatively evaluate the overall performance of the model.

Metrics	Average Line Length	Word Repetition	Word Variation	Point-of-View
Correlation between Corpus and Generated Lyrics (by Genre)	0.858	0.728	0.680	0.817

Table. 2: Quantitative results based on self-devised metrics

We obtain a relatively strong positive correlation (> 0.6) for each of the four metrics (Table. 2), which demonstrates that our model has successfully captured some unique characteristics pertaining to each genre.

8. Discussion and Learnings

The Loss vs. Step plot (Fig. 16) suggests that the training was successful, as both the training and validation loss decreases with steps, and there doesn't seem to be a noticeable overfitting or underfitting problem here. We believe that the model is performing relatively well based on the results and the training curve.

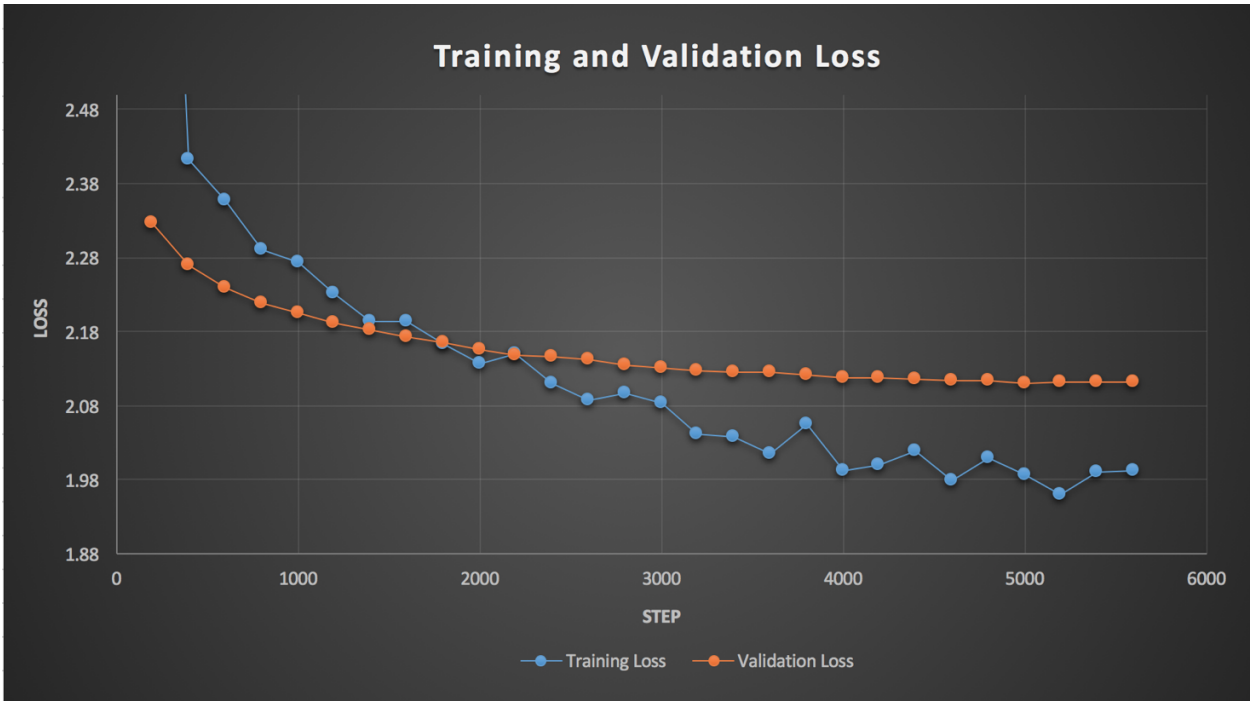


Fig. 16: Training and validation loss over steps

Discussion

As elaborated in section 6 and 7, our lyrics generation model successfully captured some distinct linguistic features of the training corpus of lyrics, and the generated lyrics are semantically apt in their relevant genre. The quality of the generated lyrics are somewhat beyond our expectation, and arguably better than those obtained in the previous study [2].

Learning

Introducing new special tokens is effective in specifying the genre as an input to the model, and we believe that it is one of the most crucial steps for text generation models to produce better results. In addition, when experimenting with the text-to-image models to generate the single cover artwork, we observe that directly passing the bulk of lyrics to the Stable Diffusion model is not a good strategy, which often leads to images of undesirable long text (Fig. 17). In contrast, a well-engineered prompt (after text summarization) works much better and produces more visually appealing images (Fig. 18).

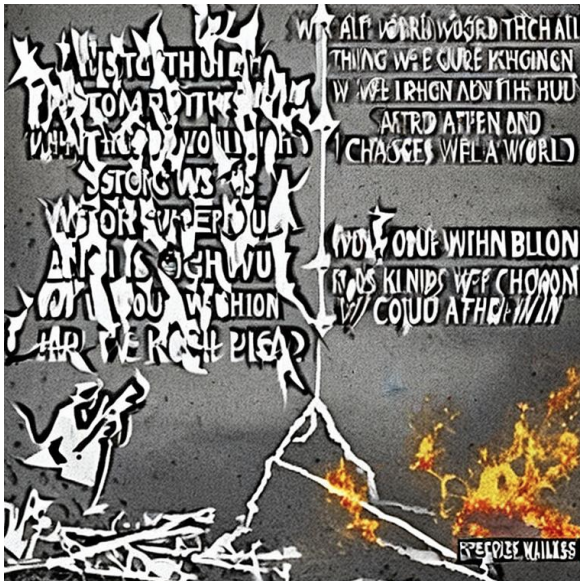


Fig. 17: Image generated using full lyrics as prompt

Fig. 18: Image generated using well-engineered prompt

Potential Improvement for Similar Projects

We observe that some of the machine-generated lyrics contain swear words and slurs. In the context of this project, some degree of profanity is expected for certain music genres (e.g. Rap). However, when starting another similar project, we will use a content filter such as moderation endpoint to assess whether the user-generated content is inappropriate, and display a warning message when such content is detected.

After data preprocessing, each training example has one special genre-specific token added at the beginning of the lyrics, and some training examples are identical except for the genre token. This is because music genres are not mutually exclusive, and a song can be classified into multiple genres. Although this doesn't seem to negatively impact the model performance, a more intuitive way to preprocess the lyrics (or any other literary texts that can be classified into multiple categories) is to allow multiple genre-specific tokens to be added to each training example. Moreover, this would also allow the model to generate lyrics with the characteristics of multiple genres.

9. Individual Contributions

Accomplishment	Contributor(s)
Search appropriate source of data	Yiqian
Collect raw data with LyricsGenius API	Yiqian, Wenzhe
Data cleaning and preprocessing	Yiqian, Wenzhe
Preliminary model training based on genre	Yiqian, Wenzhe
Fine-tuning and testing	Wenzhe
Progress Report	Yiqian, Wenzhe
Devise the metrics for quantitative analysis	Wenzhe
Implement the self-devised metrics with Python	Wenzhe
Test various text-to-text and text-to-image models	Yiqian
Write the Gradio implementation of the UI	Yiqian
Presentation Slides and Final Report	Yiqian, Wenzhe

References

[1] J. Guan, F. Huang, Z. Zhao, X. Zhu, and M. Huang, “A knowledge-enhanced pretraining model for Commonsense story generation,” *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 93–108, 2020.

[2] Gill, H., Marwell, N., & Lee, D., “Deep Learning in Musical Lyric Generation: An LSTM-Based Approach,” *The Yale Undergraduate Research Journal*, *1*(1), 2020.

Appendix

Link to the Final Application: https://huggingface.co/spaces/ECE1786-AG/ArtIstic-GENREator

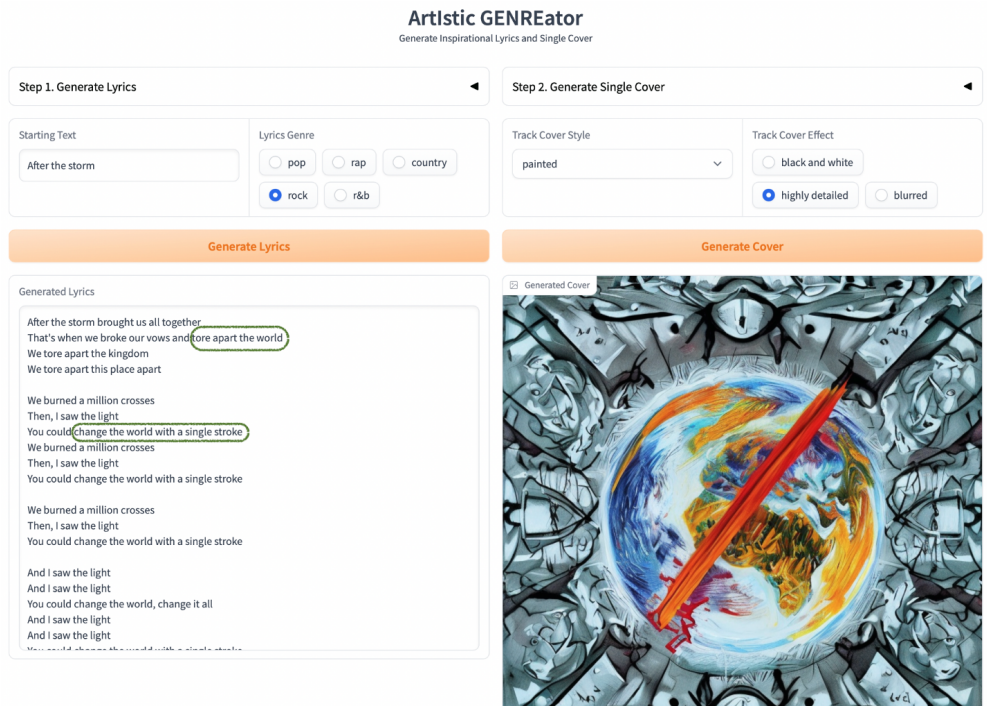


Fig. 19: Sample Lyrics and Cover Art ("After the storm…", Rock)

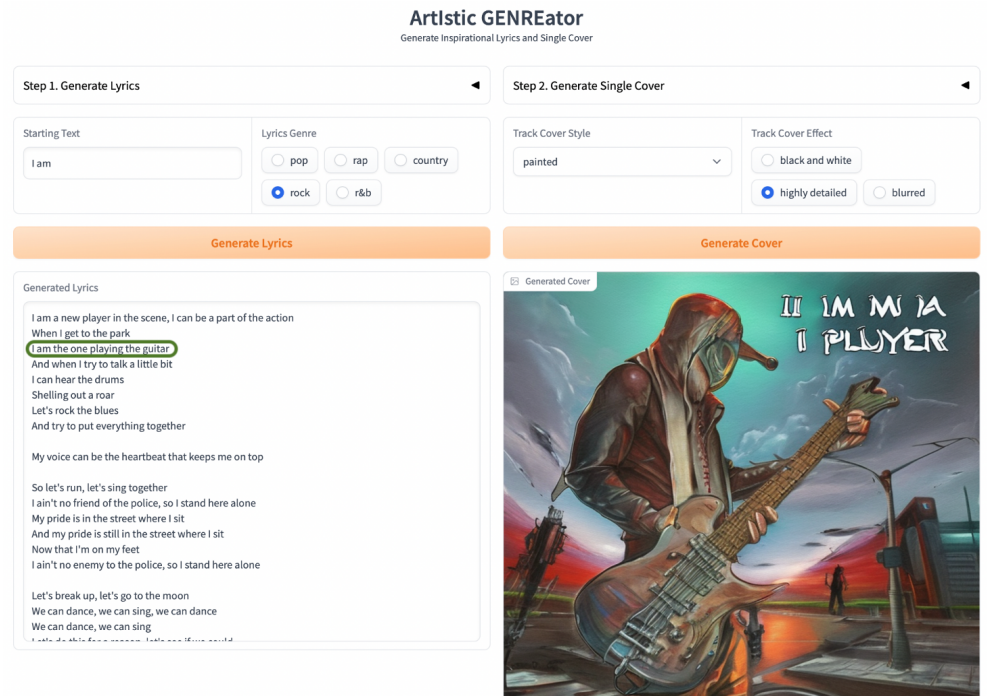


Fig. 20: Sample Lyrics and Cover Art ("I am…", Rock)