Lyric Genie 🧞

Predicting Song Popularity
&
Building Artist-Specific Lyric Generators

Team 5
Individual Report

Jiwoo Suh

Table of Contents

Phase 2 - Lyric Analysis & Lyric Generation

1. Introduction

In the phase 2 of the project, the objective was to develop a Lyric Analyzer and Lyrics Generator focused on English songs using a track dataset from Spotify. The primary goals included summarizing lyrics and extracting keywords using pre-trained models, contributing to the understanding and analyzing song content.

2. Data Preprocess

2.1. Data Preprocessing for Lyric Analysis:

The initial step in developing a Lyric Analyzer involves preprocessing the data to ensure a clean and focused dataset. The following steps outline the preprocessing procedures applied to the Spotify track dataset:

1) Get all the English lyrics from the dataset
2) Get related features: The dataset for the lyric analysis has been meticulously prepared, focusing on key columns to facilitate meaningful insights into English songs. The selected columns and their descriptions are as follows:
    - track_id (Character): Song unique ID.
    - track_name (Character): Name of the song.
    - track_artist (Character): Artist(s) associated with the song.
    - lyrics (Character): Lyrics for the song, providing the textual content for analysis.

2.2. Data Preprocessing for Lyric Generator:

The dataset for the lyric generator needs additional processing to obtain the top 10 artists with the most lyrics. We sorted artists by the number of songs they have in our dataset. Based on that, these are the artists selected for the modeling, and the number of songs.

Artist	Number of Songs
Queen	122
David Guetta	70
Drake	65
Guns N' Roses	63
Logic	63
The Chainsmokers	59
Martin Garrix	52
2Pac	51

The Weeknd	49
Eminem	45

We made training and validation datasets for each artist with their lyrics data. So there are 10 training dataset and 10 validation dataset in total.

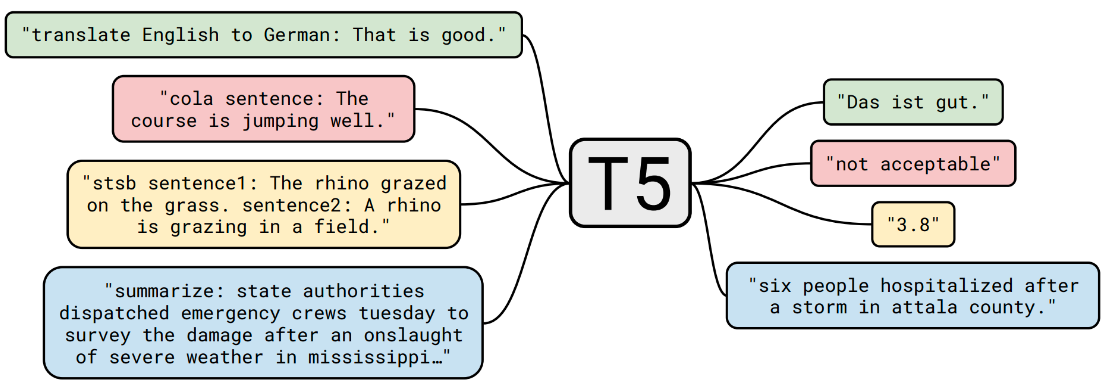
3. Models

3.1. Model 1: Lyric analysis - Summarization & Keyword Extractor

3.1.1. Lyric Summarization

To summarize the lyric data, “Falconsai/text_summarization” model has been used. The Fine-Tuned T5 Small model represents a specialized variant of the T5 transformer model tailored for the task of text summarization.

3.1.1.1. Fine-Tuned T5 Small for Text Summarization



The T5 (Text-to-Text Transfer Transformer) framework is depicted in the diagram above, illustrating a unified approach for various tasks such as translation, question answering, and classification. In this text-to-text framework, the model is trained to generate target text from input text, enabling a consistent use of the same model, loss function, hyperparameters, etc., across diverse tasks. This unified approach serves as a standard testbed for evaluating methods within the empirical survey, showcasing the versatility and efficiency of the T5 model.

Specifically named "t5-small," this model is adept at producing succinct and coherent summaries from input text. The underlying architecture is pre-trained on a diverse corpus of text data, enabling it to grasp essential information and generate meaningful summaries.

Fine-tuning details are below:

- Architecture: T5 transformer model (variant: "t5-small").
- Pre-training: Conducted on a diverse text corpus to capture essential information.
- Fine-tuning Hyperparameters:
  - Batch Size: 8 for efficient computation and learning.

- Learning Rate: 2e-5 for a balance between convergence speed and optimization.

3.1.1.2. Implementation and Limitations

The reason I selected this model for lyric summarization is that the intended use of this model is for generating concise and coherent text summaries. It is suitable for applications involving summarization of lengthy documents, news articles, and textual content.

The process involves loading a dataset and applying the summarization model to generate concise summaries. While proficient in general text summarization, the model may encounter challenges with highly specialized or domain-specific content because of the characteristics of lyrics.

3.1.2. Keyword Extraction

To extract the keywords from lyric data, “KeyBERT” model has been used.

3.1.2.1. KeyBERT

KeyBERT is a lightweight and accessible keyword extraction method that utilizes BERT embeddings to identify keywords and keyphrases most representative of a given document. Unlike other methods such as Rake, YAKE!, and TF-IDF, KeyBERT offers simplicity and effectiveness in extracting relevant keywords from textual data.

The KeyBERT approach involves three main steps:

1) Document Embeddings Extraction: Utilizing BERT, embeddings are extracted to create a document-level representation, capturing the overall context of the lyrics.
2) N-gram Words/Phrases Embeddings: Word embeddings are obtained for N-gram words and phrases within the document.
3) Cosine Similarity Calculation: Simple cosine similarity is applied to identify words and phrases that are most similar to the document, thereby determining the keywords that best describe the entire content.

KeyBERT stands out as a quick and easy-to-implement solution, requiring minimal lines of code and no need for training from scratch. While various methods exist for keyword generation, the selection of KeyBERT for this project is motivated by its simplicity, effectiveness, and utilization of BERT embeddings. BERT, a state-of-the-art language model, ensures that the generated keywords capture the nuanced meanings embedded in song lyrics.

3.1.2.2. Implementation and Limitations

To demonstrate the application of KeyBERT for keyword extraction in the context of song lyrics, a Python function named get_keywords has been developed. This function takes lyrics as input and returns a list of extracted keywords. The parameters keyphrase_ngram_range and stop_words allow for customization of the extraction process. While KeyBERT offers simplicity and efficiency for keyword extraction in song lyrics, it comes with certain limitations. The tool may struggle to capture the nuances of domain-specific language found in lyrics, particularly when dealing with poetic expressions and cultural references. Its generalization might overlook the complexity of sentence structures, rhyme schemes, and specific themes in songs. Additionally, the heavy reliance on pre-trained BERT embeddings may hinder accuracy when faced with highly specialized or novel vocabulary.

3.2. Model 2: Lyric generator

For the artist-specific lyric generator, the OpenAI GPT-2 model with language model head has been fine-tuned to be trained with the custom dataset, which is the lyrics dataset, and used to generate lyrics.

3.2.1. GPT2 Language Model Head Model (GPT2LMHeadModel)

The inception of the GPT-2 model, as presented in "Language Models are Unsupervised Multitask Learners" by OpenAI, introduced a causal (unidirectional) transformer. The model underwent pretraining through language modeling on an extensive corpus of approximately 40 GB of text data. The GPT-2, comprising a staggering 1.5 billion parameters, was trained on a diverse dataset of 8 million web pages. The model's primary objective was simple yet powerful: predict the next word in a sequence given all preceding words. This straightforward task, when applied to the varied dataset, naturally encompassed demonstrations of numerous tasks across diverse domains. Notably, GPT-2 represents a significant advancement, being a direct scale-up of its predecessor GPT, featuring over 10 times the parameters and trained on more than 10 times the volume of data.[1]

The GPT-2LMHead model utilized in this project is a variant of the GPT-2 transformer model, specifically designed for language modeling tasks. The GPT-2 Model employs a transformer architecture, featuring a language modeling head positioned above it. This head consists of a linear layer with weights intricately linked to the input embeddings. In particular, the LMHead part of the model focuses on predicting the next word in a sequence given the context of preceding words. This autoregressive language model is highly effective in generating coherent and contextually relevant text, making it well-suited for lyric generation.

3.2.2. Fine-tuning

During fine-tuning, several hyperparameters were carefully selected to optimize model performance. These include:

- **Batch Size**: Set to 8 for efficient computation and learning.
- **Epochs**: Trained for 10 epochs to allow the model to capture intricate patterns in the lyrics dataset.
- **Weight Decay**: Applied with a value of 0.01 to control overfitting.

[1] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

3.2.3. Training

The training process involved splitting the dataset into training and evaluation sets with a 9:1 ratio. The GPT-2LMHead model was trained with the specified hyperparameters, and checkpoints were saved at regular intervals (every 500 steps). The use of an evaluation dataset allowed monitoring model performance and ensuring optimal training progress.

The code segment of a function generate_text leverages a pretrained GPT-2 language model to generate text based on a given input sequence. The key parameters[2] influencing the generation process are as follows:

- **top_k** (int, optional): Set as 50. If set to a value greater than 0, this parameter ensures that only the top k tokens with the highest probability are considered during the generation process, thereby implementing top-k filtering.
- **top_p** (float, optional): Set as 0.95. When set to a value less than 1.0, this parameter enforces nucleus filtering. Specifically, it retains only the top tokens with a cumulative probability greater than or equal to top_p.[3]
- **num_beams** (int, optional): Set as 5. The inclusion of beam search enhances the generation process by maintaining the most likely num_beams hypotheses at each step. This minimizes the risk of overlooking high probability word sequences and aids in selecting the hypothesis with the highest overall probability.
- **no_repeat_ngram_size** (int, optional): Set as 5. This parameter mitigates the issue of repetitive word sequences by preventing the appearance of specified n-grams. Setting no_repeat_ngram_size=5 ensures that no 5-gram appears more than once in the generated text.
- **early_stopping** (bool, optional): Set as True. When set to True, early stopping terminates the generation process when all beam hypotheses reach the end-of-sequence (EOS) token.

3.2.4. Evaluation

To assess the quality and coherence of the generated lyrics, a robust evaluation approach based on semantic similarity was employed. The evaluation leveraged a state-of-the-art sentence-transformers model, specifically 'sentence-transformers/all-mpnet-base-v2', which maps sentences and paragraphs into a 768-dimensional dense vector space, facilitating tasks such as clustering and semantic search.

- **Sentence Embeddings and Pooling:**

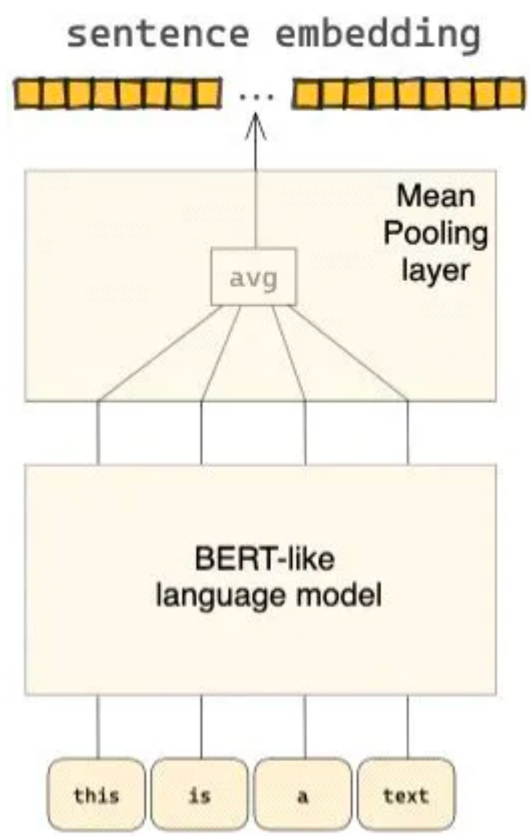
[2] https://huggingface.co/blog/how-to-generate
[3] Nucleus filtering is described in Holtzman et al. (http://arxiv.org/abs/1904.09751)

Before delving into the evaluation results, it's crucial to understand the concept of sentence embeddings. Unlike token-level embeddings, which have one embedding per token, sentence embeddings provide a single embedding for the entire sequence. The process of deriving a sentence embedding involves a technique known as "pooling," where the granular token-level representations are compressed into a fixed-length representation intended to capture the overall meaning of the entire sequence.

It's essential to note that sentence embeddings are inherently compressions of information, resulting in a lower level of granularity compared to token-level embeddings. Transformers often truncate input sequences longer than a max_seq_length, which can vary between models. This further emphasizes that sentence embeddings are representations with a reduced level of detail due to the inherent lossiness of compression.

- **Mean Pooling for Semantic Comparison:**

The evaluation employed mean pooling, one of the commonly used pooling methods. Mean pooling aggregates information by taking the element-wise arithmetic mean of token-level embeddings. This approach ensures a comprehensive representation of the semantic content of the lyrics, allowing for a nuanced assessment of the generated text.



[4]

While mean pooling was utilized in this evaluation, it's worth noting that other pooling methods, such as max pooling and mean_sqrt_len pooling, exist, although they are less

[4] https://blog.ml6.eu/the-art-of-pooling-embeddings-c56575114cf8

frequently employed. HuggingFace defaults to CLS pooling for sequence classification tasks, showcasing the absence of a universal consensus on the most suitable pooling method.

- **Cosine Similarity for Evaluation:**

The evaluation metric chosen for semantic similarity was cosine similarity. Cosine similarity measures the cosine of the angle between two vectors and is widely used for comparing the similarity of embeddings. In this context, the cosine similarity was computed between the generated lyrics and the real lyrics of each artist. The result provides a quantitative measure of how closely the generated lyrics align with the semantic content of the original lyrics.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

This rigorous evaluation strategy ensures a comprehensive understanding of the semantic coherence and fidelity of the generated lyrics, contributing valuable insights into the performance of the lyric generation model.

- **Human Judgement:**

To complement the quantitative assessment, human judgement was incorporated. The quality of the generated lyrics was subjectively assessed, with three categories: Good, Average, and Poor. A total of 20 evaluations were conducted, resulting in a distribution of 20% rated as Good, 40% as Average, and 20% as Poor. This qualitative evaluation allowed for a more holistic understanding of the model's performance, capturing aspects that might not be fully reflected in quantitative measures.

By blending both quantitative and qualitative evaluation methodologies, this approach provided a robust and multi-faceted assessment of the lyric generation model. The inclusion of human judgement added a valuable layer of insight into the perceived quality of the generated lyrics, complementing the semantic similarity assessment based on state-of-the-art sentence embeddings.

3.2.5. Implementation and Limitations

The implementation involved creating artist-specific datasets, training individual models for each artist, and subsequently generating lyrics based on user-provided sequences. The model evaluation included both human judgment and cosine similarity calculation against real lyrics, utilizing the Sentence-Transformers model for embedding comparison.

Despite its effectiveness, the GPT-2LMHead model may face challenges in capturing highly specific lyrical nuances, especially in the presence of intricate rhyme schemes or artist-specific language. Additionally, the model's generation may be influenced by biases present in the training data, requiring careful consideration of potential limitations in lyric diversity.

In conclusion, the artist-specific lyric generator demonstrates the potential of the GPT-2LMHead model for personalized content creation. The carefully tuned hyperparameters, training process, and evaluation methods contribute to a robust lyric generation system that aligns with the project's goal of providing artist-specific and contextually relevant lyrics.

4. Application - Lyric Genie

4.1. Introduction

The Lyric Genie app is a unique artist-specific lyric generator developed using the Streamlit framework. The app allows users to generate lyrics in the style of their favorite artists and provides analysis features to evaluate the similarity and quality of the generated lyrics. The key features of the app include:

4.2. Artist-Specific Lyrics Generation

Users can select an artist from a predefined list of 10 artists and input a starting sequence to generate lyrics mimicking the chosen artist's style.

👩‍🎤 Top 10 Artists by Lyrics Count

track_artist	lyrics_count
Queen	122
David Guetta	70
Drake	65
Guns N' Roses	63
Logic	63
The Chainsmokers	59
Martin Garrix	52
2Pac	51
The Weeknd	49
Eminem	45

🎵 Generate Your Hit Lyrics

Pick your artist 🎤:

The Chainsmokers

- Queen
- David Guetta
- Drake
- Guns N' Roses
- Logic
- The Chainsmokers
- Martin Garrix
- 2Pac

artist | sequence | generated_lyrics

🎵 Generate Your Hit Lyrics

Pick your artist 🎤:

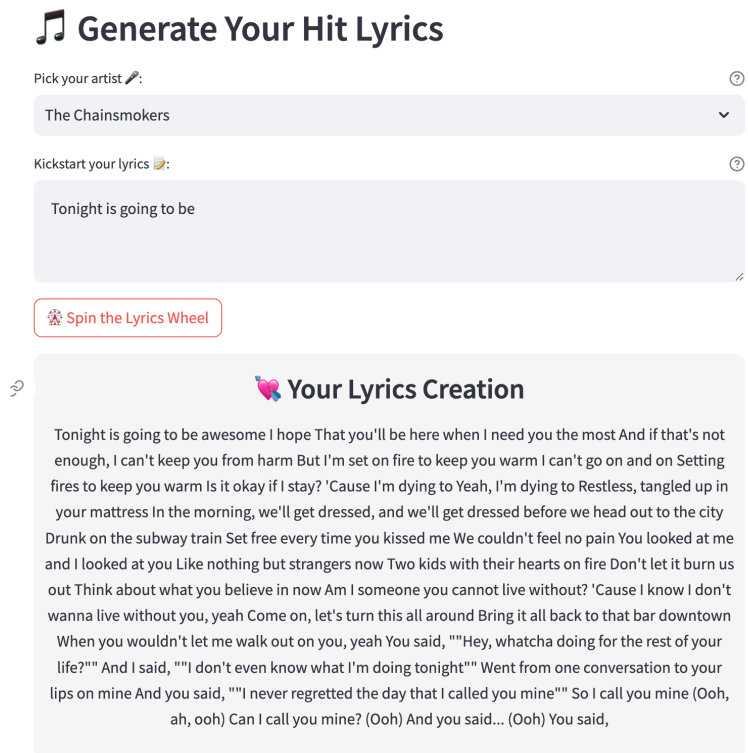
The Chainsmokers

Kickstart your lyrics 📝:

Tonight is

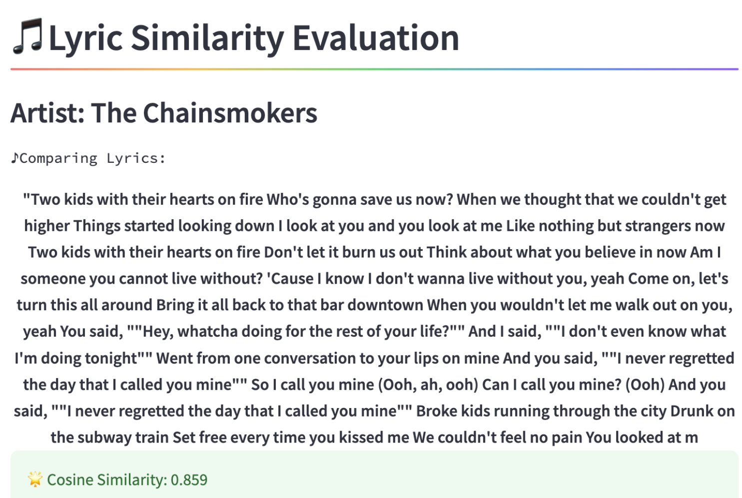
Press ⌘+Enter to apply

🎡 Spin the Lyrics Wheel



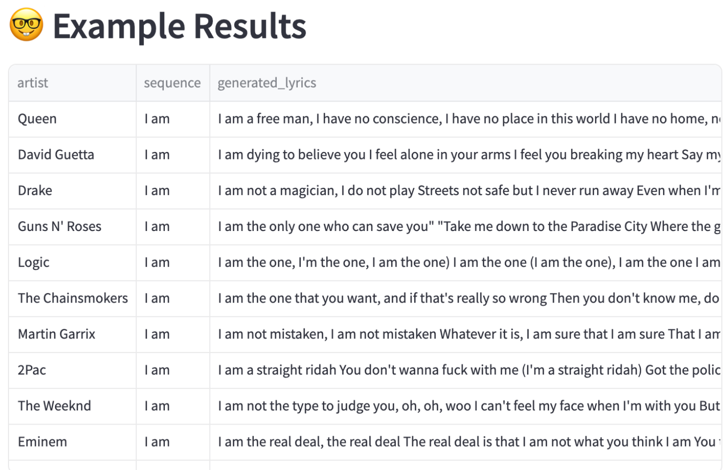
4.3. Lyric Similarity Evaluation

The app calculates and displays the cosine similarity between the generated lyrics and real lyrics from the chosen artist's dataset.



4.4. Example Results

Example results from the 10 models with different input sequence. You can compare the various lyrics with the same input sequence by the artists.

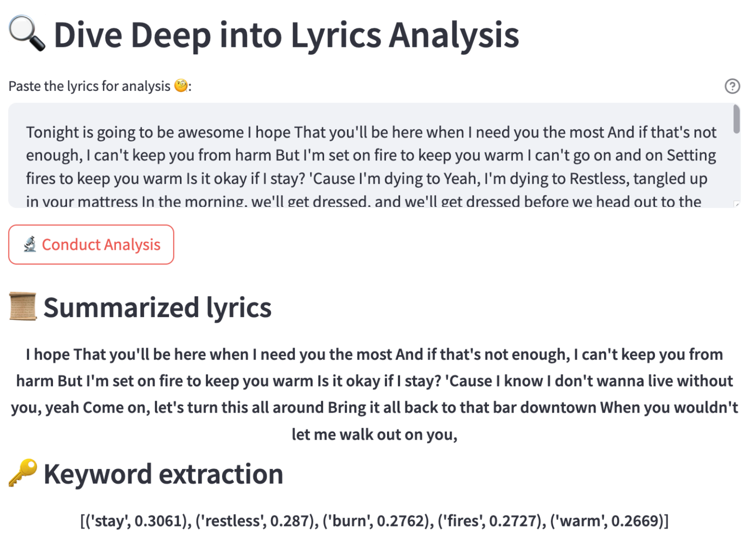


🤓 Example Results

artist	sequence	generated_lyrics
Queen	I am	I am a free man, I have no conscience, I have no place in this world I have no home, n
David Guetta	I am	I am dying to believe you I feel alone in your arms I feel you breaking my heart Say my
Drake	I am	I am not a magician, I do not play Streets not safe but I never run away Even when I'm
Guns N' Roses	I am	I am the only one who can save you" "Take me down to the Paradise City Where the g
Logic	I am	I am the one, I'm the one, I am the one) I am the one (I am the one), I am the one I am
The Chainsmokers	I am	I am the one that you want, and if that's really so wrong Then you don't know me, do
Martin Garrix	I am	I am not mistaken, I am not mistaken Whatever it is, I am sure that I am sure That I am
2Pac	I am	I am a straight ridah You don't wanna fuck with me (I'm a straight ridah) Got the polic
The Weeknd	I am	I am not the type to judge you, oh, oh, woo I can't feel my face when I'm with you But
Eminem	I am	I am the real deal, the real deal The real deal is that I am not what you think I am You

4.5. Lyrics Analysis

Users can input lyrics for analysis, which includes summarization and keyword extraction.



5. Conclusion

In this phase, we embarked on a comprehensive exploration of lyric analysis and generation, delving into two key models: a lyric summarization and keyword extractor, and a lyric generator. Our journey commenced with meticulous data preprocessing, ensuring the quality and relevance of the lyrical content. This phase aimed to unravel the intricacies of lyrical data and harness the potential of state-of-the-art models for music-related tasks.

6. Furthermore

6.1. Further Improvements

The current implementation of the application has laid a solid foundation, and there are several avenues for enhancement and refinement. The following are key areas for further improvement:

- Performance Optimization through Fine Tuning: Adjusting hyperparameters and training on additional relevant data to achieve better accuracy and responsiveness.
- Experimenting with state-of-the-art models like "llama" and other cutting-edge language models can provide insights into their effectiveness in lyric analysis. Assessing their performance may lead to the adoption of more advanced models.
- Multilingual Lyrics Handling:
  Extend the application's capabilities to handle lyrics from multiple languages. This involves investigating and implementing alternative models designed specifically for multilingual text processing. By doing so, the application can seamlessly analyze lyrics in various linguistic contexts, catering to a more diverse user base.
- Data Enrichment from Lyrics Websites:
  Enhance the dataset used for training by incorporating additional lyrics data from reputable sources such as Genius. Regularly updating the dataset ensures that the model remains current and capable of handling a diverse range of musical genres and styles. This continuous enrichment contributes to the accuracy and relevance of lyric analysis.
- Integration with Text-to-Speech Models:
  Investigate the possibility of integrating a text-to-speech (TTS) model to generate speech based on the analyzed lyrics. This innovative feature would enable users to experience the synthesized output in the voice characteristic of each artist, providing a more immersive and personalized experience. The integration of TTS models adds a new dimension to the application's capabilities, making it more engaging for users.

These proposed improvements aim to elevate the model and application’s performance, broaden its language support, leverage advanced models, enrich the dataset, and introduce innovative features. Continuous exploration and refinement in these areas will contribute to a more robust and versatile lyric analysis application. As the field of natural language processing evolves, incorporating these enhancements will ensure that the Lyric Genie app remains at the forefront of lyric analysis technology.

7. Calculated Code portion

Approximately, the 60% of codes are written by me or modified by me.

8. References

1. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer https://jmlr.org/papers/volume21/20-074/20-074.pdf
2. Sharma, P., & Li, Y. (2019). Self-Supervised Contextual Keyword and Keyphrase Retrieval with Self-Labelling.
3. https://github.com/huggingface/transformers/tree/main/examples/pytorch/language-modeling#gpt-2gpt-and-causal-language-modeling
4. https://huggingface.co/blog/how-to-generate