Lyric Genie

Predicting Song Popularity
&
Building Artist-Specific Lyric Generators

Team 5

Jiwoo Suh
Sanjana Godolkar
Upmanyu Singh

Table of Contents

Phase 1 - Song Popularity Prediction

1. Introduction

In Phase 1 of our project titled "Lyrics Genie," we've developed regression and classification models to predict the popularity of songs, employing Natural Language Processing (NLP) to analyze lyrical content. Our approach bifurcates into predicting a song's popularity score via regression and classifying tracks into popularity brackets using classification. This report encapsulates our methodologies—from data preprocessing and lyrical vectorization to model selection and evaluation—laying the groundwork for the project's subsequent phases that will delve into broader song attributes and advanced NLP techniques.

2. Data Preprocessing

The dataset for our NLP project is a comprehensive collection of 15,405 songs, meticulously curated from an initial set of 18,454 tracks. It was refined by removing non-English songs, resulting in a dataset with 22 key features. This dataset offers a rich blend of categorical and continuous data types, encompassing both lyrical content and a spectrum of auditory features. The categorical variables include `track_name`, `track_artist`, `lyrics`, `track_album_name`, `track_album_release_date`, `playlist_name`, `playlist_genre`, and `playlist_subgenre`. Continuous variables include auditory features such as `danceability`, `energy`, `key`, `loudness`, `mode`, `speechiness`, `acousticness`, `instrumentalness`, `liveness`, `valence`, `tempo`, and `duration_ms`, with `key` and `mode` as integer values. The `language` attribute ensures uniformity in language for effective NLP processing.

Statistically, the dataset provides a broad perspective on the music industry. The mean track popularity score is 41.87 with a standard deviation of 24.67, indicating a wide range of song popularity. Other auditory features exhibit moderate to high averages, like danceability (0.632) and energy (0.689). The dataset also boasts a rich diversity in its categorical fields, with over 12,500 unique track names and nearly 5,000 unique artists, showcasing a wide spectrum of the musical landscape. This dataset serves as a solid foundation for our project, enabling the application of machine learning models for song popularity prediction, as well as NLP models for generating and summarizing lyrics and assessing the popularity of these generated contents.

3. Description of the NLP model & Algorithms used:

Classification:

1. Embedding:

The NLP model utilizes BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art model known for its deep understanding of context in text. BERT, developed by Google, transformed the field of NLP through its unique architecture that allows it to capture bidirectional context, meaning it can understand the meaning of a word based on all of its surroundings (both left and right context). This is crucial for tasks like sentiment analysis or summarization, where the meaning of words can drastically change based on their context.

For this project, we employed a variant of BERT called `prajjwal1/bert-tiny`, for embedding lyrics. This model is a smaller and more computationally efficient version of the original BERT, making it suitable for projects with resource constraints. The embeddings generated by this model capture the semantic nuances of the lyrics, which are crucial for tasks like summarization and generating new lyrics. The process involves tokenizing the lyrics and then passing them through the BERT model to obtain dense vector representations (embeddings) that capture the contextual meanings of words and phrases within the lyrics.

2. Model:

The model we have used in classification involves using XGBoost, an efficient and scalable implementation of gradient boosting. XGBoost stands out for its performance and speed, especially in structured or tabular data. It is particularly effective for classification and regression tasks in various fields.

XGBoost is employed to classify songs into different popularity classes based on a combination of lyrics embeddings and other song features like danceability, energy, and loudness. This classification provides insights into what makes a song popular, considering both its musical attributes and lyrical content. The use of PCA (Principal Component Analysis) for dimensionality reduction further enhances the model's efficiency by reducing the number of input features while retaining the most significant variance in the data.

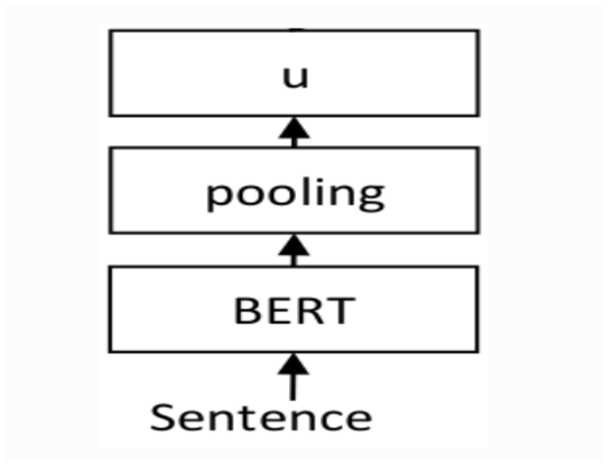
Regression:

1. Embedding:

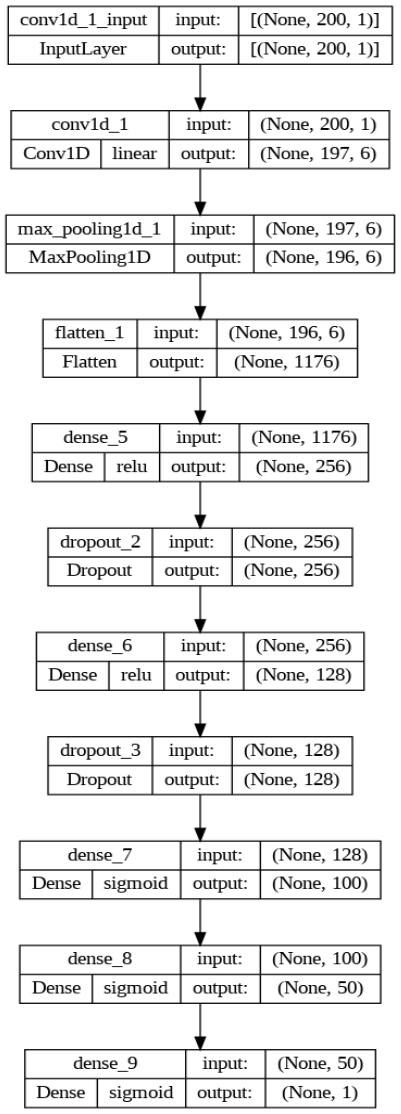
- For the analysis of song lyrics, advanced text processing techniques were employed, specifically using the ”all-mpnet-base-v2” sentence transformer. Sentence Transformers are an adaptation of traditional BERT-like Transformer models, designed to produce meaningful sentence embeddings. The core idea is that sentences with similar meanings are positioned closer together in the embedding space, which is highly beneficial for tasks like semantic search, clustering, and similarity assessment.

- MPNet: Stands for ”Masked and Permuted Pre-training for Language Understanding”, a pre-training method that expands upon the architectures of BERT and XLNet. MPNet is specifically designed to capture bidirectional contexts by learning to predict both masked and permuted tokens, thereby enhancing its capability to understand and represent language.

- all-mpnet-base-v2: This model is a fine-tuned variant of MPNet, tailored to generate universally applicable embeddings across various domains and applications. The model’s adaptability stems from its training on a wide array of data sources and tasks.



2. Model:


conv1d_1_input | input: | [(None, 200, 1)]
InputLayer | output: | [(None, 200, 1)]
conv1d_1 | input: | (None, 200, 1)
Conv1D | linear | output: | (None, 197, 6)
max_pooling1d_1 | input: | (None, 197, 6)
MaxPooling1D | output: | (None, 196, 6)
flatten_1 | input: | (None, 196, 6)
Flatten | output: | (None, 1176)
dense_5 | input: | (None, 1176)
Dense | relu | output: | (None, 256)
dropout_2 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_6 | input: | (None, 256)
Dense | relu | output: | (None, 128)
dropout_3 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_7 | input: | (None, 128)
Dense | sigmoid | output: | (None, 100)
dense_8 | input: | (None, 100)
Dense | sigmoid | output: | (None, 50)
dense_9 | input: | (None, 50)
Dense | sigmoid | output: | (None, 1)

4. Experimental Setup:

Data Preparation

The dataset used in our project encompasses a wide range of features, including song attributes (like danceability, energy, and tempo), textual content (lyrics), and popularity metrics. Initially, the dataset was preprocessed to ensure quality and relevance. This involved:

1. Data Cleaning: Dropping duplicates based on 'track_name', 'track_artist', and 'lyrics.1' to ensure uniqueness.
2. Feature Selection: Selecting a mix of numerical (such as 'danceability', 'energy') and categorical features ('track_artist', 'playlist_genre') for the model.
3. Data Transformation:
   - Numerical Features: Standardized using `StandardScaler` to have zero mean and unit variance.
   - Categorical Features: Transformed using `OneHotEncoder` to convert them into a format suitable for model input.
   - Lyrics Processing: Lyrics were tokenized and embedded using BERT to capture the contextual information in a dense vector form.
4. Target Variable Transformation: 'track_popularity' was categorized into four classes for classification purposes.

Apart from that, there were some other features which we added in the dataset which were based on the lyrics:

- Syllables Per Line (syllables_per_line): Calculates the total number of syllables in each line of the lyrics.
- Syllables Per Word (syllables_per_word): Computes the average number of syllables per word in the lyrics.
- Syllable Variation (syllable_variation): Measures the variation in the number of syllables across lines in the lyrics.
- Novel Word Proportion (novel_word_proportion): Calculates the proportion of new words in each line compared to the previous line.
- Rhymes Per Line (rhymes_per_line): Counts the number of rhyming words in each line.
- Rhymes Per Syllable (rhymes_per_syllable): Determines the ratio of rhyming words to total syllables.
- Rhyme Density (rhyme_density): Computes the density of rhymes in the lyrics.
- End Pairs Per Line (end_pairs_per_line): Calculates the percentage of consecutive lines ending in rhyming words.
- End Pairs Variation (end_pairs_variation): Analyzes how the number of syllables in the last word of a line changes across consecutive lines.
- Average End Score (average_end_score): Measures the phonetic similarity of the ending words in consecutive lines.
- Average End Syllable Score (average_end_syl_score): Calculates the average phonetic similarity score of the last syllables in consecutive lines.
- Rhyme Length Count (count_rhyme_lengths): Counts the distribution of rhyme lengths in the lyrics.

Performance Evaluation

Performance evaluation is crucial to assess the effectiveness of the model. In our project, the following methods were used:

1. Splitting Strategy: The dataset was split into training (80%), validation (10%), and testing (10%) sets. This ensures a comprehensive evaluation across unseen data.
2. Accuracy Measurement: The primary metric for evaluating model performance was accuracy, which provides a straightforward understanding of the model's effectiveness in classifying songs into the correct popularity class.
3. Classification Report: A detailed report including precision, recall, and F1-score for each class was generated. This provides deeper insights into the model's performance, especially in handling class imbalances.

The experimental setup aims to create a robust model that accurately predicts song popularity based on a variety of features, with a focus on ensuring the model's applicability to real-world scenarios and unseen data.

5. HyperParameters:

In the experimental setup for our project, hyperparameter tuning and strategies to prevent overfitting and extrapolation are critical components. Let's break down each aspect:

Regression:

Number and Size of Convolutional and Dense Layers: The model includes a Conv1D layer and several Dense layers with varying numbers of neurons (512, 256, 128, 100, 50). Adjusting the number of layers and the number of neurons in each layer can significantly impact the model's capacity to learn from data.

Activation Functions: You have used relu for the initial dense layers and sigmoid for the latter layers and the output layer. Experimenting with different activation functions (like tanh, softmax, etc.) could be part of the tuning process.

Regularization: The Conv1D layer uses L2 regularization (kernel_regularizer='l2'). Tuning might involve experimenting with the regularization rate or trying different regularization techniques like L1 regularization or Dropout.

Learning Rate of the Optimizer: The Adam optimizer is used with a learning rate of 0.0001. Tuning the learning rate is a critical aspect, as it determines the size of the steps the optimizer takes during training.

Loss Function: The model uses Mean Squared Error (mse) as the loss function. Although mse is common for regression tasks, tuning might involve trying other loss functions like Mean Absolute Error (mae).

Commented-Out Layers and Regularization: Several layers and dropout regularizations are commented out (#). Part of tuning could involve experimenting with including or excluding these layers and dropout regularizations to see how they affect model performance.

Kernel Size and Strides in Conv1D Layer: The Conv1D layer is set with specific kernel size and padding. Adjusting these can impact how the model processes input data.

Classification:

Our project utilized an XGBoost classifier, a robust and efficient machine-learning algorithm. Key hyperparameters that were tuned for optimal performance include:

1. Learning Rate: Controls the step size at each iteration while moving toward a minimum of a loss function. A smaller learning rate may lead to a better model but can require more iterations to train.
2. Max Depth: Sets the maximum depth of a tree. Deeper trees can model more complex patterns but also increase the risk of overfitting.
3. Number of Estimators: Refers to the number of trees in the forest. Typically, more trees increase performance but also computational complexity.
4. Subsample: The fraction of samples to be used for fitting each tree. Using a subset can prevent overfitting but too small a fraction might lead to underfitting.
5. Objective: Since this is a classification problem, 'multi:softmax' was used for a multi-class classification.

These hyperparameters were fine-tuned using grid search in conjunction with cross-validation.

Detecting and Preventing Overfitting and Extrapolation

1. Train-Validation-Test Split: The data was divided into training, validation, and test sets. This separation allows for the evaluation of model performance on unseen data, a key indicator of overfitting.

2. Cross-Validation: Implementing cross-validation during model training helps in assessing the model's ability to generalize to an independent dataset.

3. Regularization: XGBoost has built-in L1 (Lasso regression) and L2 (Ridge regression) regularization which helps in reducing overfitting by penalizing complex models.

4. Early Stopping: A technique where the training process is stopped as soon as the performance on the validation set starts to degrade. This is crucial in preventing overfitting.

5. Model Complexity: By controlling the depth of the trees and the number of trees, the complexity of the model can be managed. A less complex model is less likely to overfit.

6. Performance Metrics: Monitoring metrics such as accuracy, precision, recall, and F1-score for both training and validation sets helps in identifying overfitting. A large discrepancy in performance on training versus validation data is a red flag.

7. PCA for Dimensionality Reduction: Reducing the number of features through PCA helps in mitigating overfitting as it lessens the chance of the model learning noise in the data.

8. Monitoring Learning Curves: Analyzing the learning curves that plot training and validation errors over time can be insightful. If training and validation errors diverge significantly, it might indicate overfitting.

By considering these factors in the experimental setup, our project aims to build a model that not only fits the training data well but also generalizes effectively to new, unseen data. This balance is critical for the successful application of the model in real-world scenarios.

Results:

Regression:

1. We have used diffferent metric like r-squared, adjusted r-squared, mean squared errors and mean absolute error. For the best model we got R-squared - 0.23, Adjusted R-squared - 0.17, MSE - 0.0491, MAE - 0.17.
2. There were other metrics we used like analysing plots on residuals.
   a. **Interpretation of the provided Q-Q Plot:**
      i. Center of the Plot: The points in the middle section of the plot (around the 0 quantile) follow the red line quite closely, indicating that the middle values of your data are normally distributed.
      ii. Tails of the Plot: However, as you move towards the ends (tails) of the plot, the points deviate from the red line. This indicates that the extreme values in your data set are not as normally distributed—they have heavier or lighter tails than would be expected in a normal distribution.
      iii. The left tail (the beginning of the dataset) shows that the low values in the data set are more extreme than what would be expected in a normal distribution (they lie further from the red line).The right tail shows a similar pattern, where the high values in the dataset are higher than what would be expected under normality (the points curve upwards).
   b.
   c. Fitted Values (X-axis): These are the predicted values generated by the model.
   d. Residuals (Y-axis): These are the differences between the observed values and the fitted values. A residual is calculated as Residual = Observed value - Fitted value.
   e. Interpretation:
   f. · Horizontal Distribution Around Zero: Ideally, residuals should be randomly scattered around the horizontal axis (y=0). If the residuals are equally distributed above and below the dashed

horizontal line (y=0), it suggests that the model does not have systematic errors. In the plot provided, the residuals do appear to center around zero, which is a good sign.

g. · Pattern in Residuals: In a well-fitting model, there should be no discernible pattern or shape in the scatter plot; the points should be randomly dispersed. In this plot, there seems to be a fan or cone-shaped pattern (wide on one side and narrow on the other), indicating that the variance of the residuals is not constant (a phenomenon known as heteroscedasticity). The residuals seem to spread out more for higher fitted values.

h. · Outliers: Any significant outliers would appear as points far away from the bulk of the data. In this plot, there don't appear to be any extreme outliers, but there is a slight trend of the residuals becoming more negative as the fitted values increase.

i. Interpretation of the Provided Histogram:

j. Central Peak: The histogram has its highest peak around the zero, which is a good sign indicating that the residuals have a mean around zero.

k. Shape: The shape of the histogram does not perfectly resemble a normal distribution. It appears slightly skewed to the right, with more data points having a small positive residual than a small negative residual.

l. Outliers and Skewness: The presence of more frequent higher positive residuals suggests that the model might be under-predicting more often than over-predicting. This could also indicate potential outliers or skewness in the underlying data.

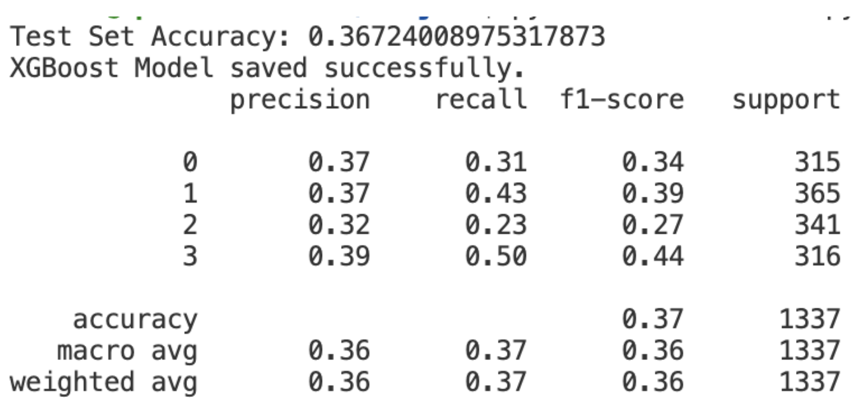
Classification:

Our experimental results are pivotal in understanding the effectiveness of our NLP model in predicting song popularity. This section details these outcomes, using a combination of statistical data, figures, and tables.

Test Set Accuracy

- Figure/Table: A table or bar graph displaying the test set accuracy of our model.
- Result: The XGBoost model achieved an accuracy of approximately 36.72% on the test set.
- Explanation: This metric is crucial as it represents the proportion of correctly predicted instances out of the total instances in the test dataset. It provides a straightforward measure of the model's performance.

Classification Report



```
Test Set Accuracy: 0.36724008975317873
XGBoost Model saved successfully.
              precision    recall  f1-score   support

           0       0.37      0.31      0.34       315
           1       0.37      0.43      0.39       365
           2       0.32      0.23      0.27       341
           3       0.39      0.50      0.44       316

    accuracy                           0.37      1337
   macro avg       0.36      0.37      0.36      1337
weighted avg       0.36      0.37      0.36      1337
```

- Figure/Table: A detailed table showing precision, recall, and F1-scores for each class.
- Result: The classification report reveals varied performance across different classes. The precision scores (the accuracy of positive predictions) for classes 0, 1, 2, and 3 were 37%, 37%, 32%, and 39% respectively. The recall scores (the ability of the classifier to find all the positive samples) were 31%, 43%, 23%, and 50% respectively. The F1-scores, which combine precision and recall into a single metric, were 34%, 39%, 27%, and 44% for each class, respectively.
- Explanation: This report provides a more detailed insight into the model's performance, breaking down its effectiveness in correctly identifying each class. The higher recall for class 3 suggests better identification of highly popular songs, while lower scores in other classes indicate areas for improvement.

Discussion

- Key Observations: The moderate overall accuracy indicates that while the model has learned to some extent to classify songs based on popularity, there's significant room for improvement. The variance in performance across different classes suggests that certain song characteristics may be more predictive of popularity than others.
- Limitations and Challenges: The relatively low precision and recall in some classes could be attributed to the inherent complexity of the task, potential biases in the dataset, or limitations in the feature set used for training the model.
- Future Work Recommendations: Enhancements could include experimenting with different feature sets, employing more sophisticated NLP techniques for lyrics analysis, or exploring more complex models and ensemble methods.

7.Summary and Conclusions:

Summary of Results

In our NLP project, we embarked on a challenging yet insightful journey to predict song popularity using classification and regression models. The core of our project involved processing and analyzing various features, including lyrics, danceability, energy, and several others, to understand their impact on song popularity. Our primary model, an XGBoost classifier, demonstrated a test set accuracy of approximately 36.72%, indicating a moderate level of predictive ability.

Key Learnings

1. Complexity of Predicting Popularity: The project underscored the intricate nature of song popularity, which is influenced by a myriad of factors, both quantitative and qualitative. Our model's varied performance across different popularity classes highlighted the challenges in capturing the nuanced elements that contribute to a song's success.

2. Importance of Feature Selection: We observed that certain features had more predictive power than others. This insight is crucial for future model improvements, as it emphasizes the need for careful feature selection and possibly the inclusion of more context-specific variables.

3. NLP and Lyrics Analysis: The use of NLP techniques for lyrics analysis was a significant aspect of our project. While it added depth to our model, it also revealed the complexities involved in accurately interpreting and quantifying lyrical content.

Suggested Improvements

1. Advanced Model Architectures: Exploring more sophisticated machine learning models, such as deep learning algorithms, could potentially enhance predictive accuracy. Implementing ensemble methods or neural network-based approaches may offer a more nuanced understanding of the data.

2. Richer Feature Engineering: Incorporating additional features, such as artist popularity metrics, historical chart performance, or even sentiment analysis of lyrics, could provide a more comprehensive view. Moreover, refining the way we process and utilize lyrics could yield better results.

3. Dataset Expansion and Diversification: Expanding the dataset to include a wider range of songs from various genres, languages, and time periods could improve the model's generalizability. This would also help in reducing potential biases in the dataset.

4. Hyperparameter Optimization: Further tuning of model hyperparameters through methods like grid search or random search could enhance model performance. Regularization techniques might also be employed to prevent overfitting.

5. Cross-validation Techniques: Implementing robust cross-validation strategies would provide a clearer picture of the model's performance and stability across different subsets of data.

Concluding Remarks:

This project has been a profound learning experience, offering valuable insights into the complexities of NLP and predictive modeling in the context of music analysis. While our current model presents a baseline, the potential for improvement and expansion is vast. Future iterations of this project could significantly benefit from the suggested enhancements, potentially leading to more accurate and reliable predictions of song popularity.

8. References

Below are the references to the online resources and models that were utilized
or adapted in this project:
1. Spotify Popularity Predictor GitHub Repository: https://github.com/
mcozar99/Spotify_Popularity_Predictor
2. Prosodic Library GitHub Repository: https://github.com/quadrismegistus/
prosodic
3. Sentence Transformer Model - all-mpnet-base-v2 on Hugging Face: https:
//huggingface.co/sentence-transformers/all-mpnet-base-v2
4. BERT Base Uncased Model on Hugging Face: https://huggingface.
co/bert-base-uncased
5. BERT Tiny Model on Hugging Face: https://huggingface.co/prajjwal1/
Bert-tiny
6.Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
7.Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System.
8.Online resources and documentation for Python libraries (Pandas, Scikit-Learn, XGBoost, Py- Torch, Transformers), and Streamlit.

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.

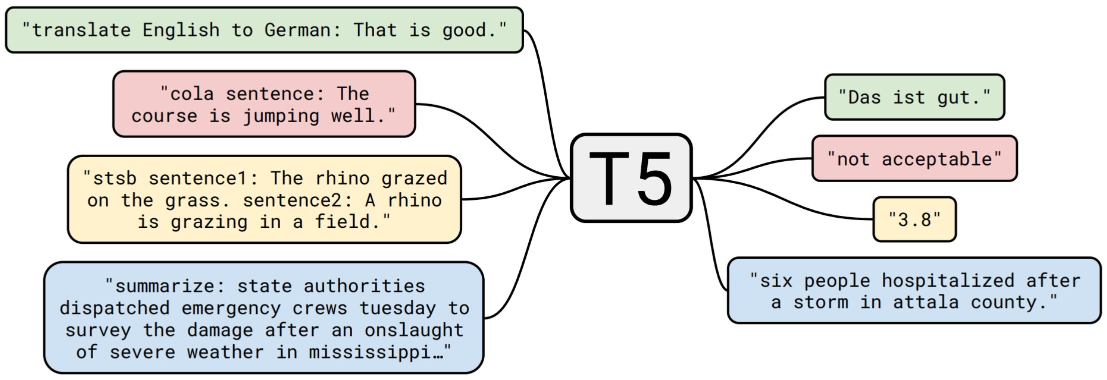
3. Models

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


"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

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.

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.

[1] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.
[2] https://huggingface.co/blog/how-to-generate
[3] Nucleus filtering is described in Holtzman et al. (http://arxiv.org/abs/1904.09751)

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.

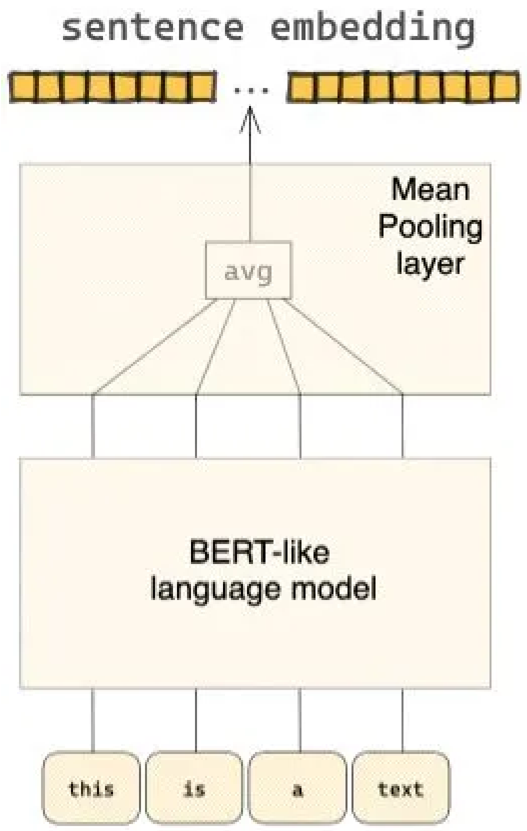
- **Sentence Embeddings and Pooling:**

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 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.

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

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

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.

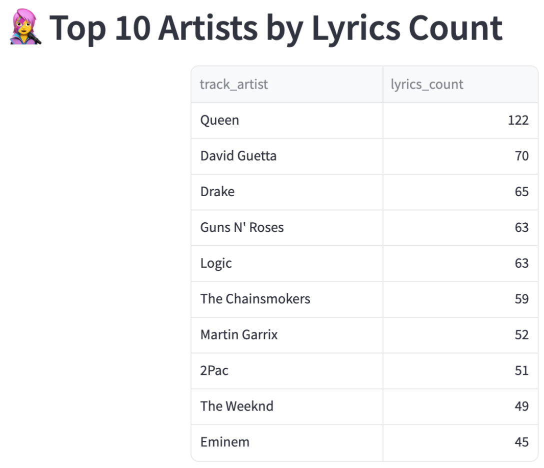
4. Application - Lyric Genie

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:

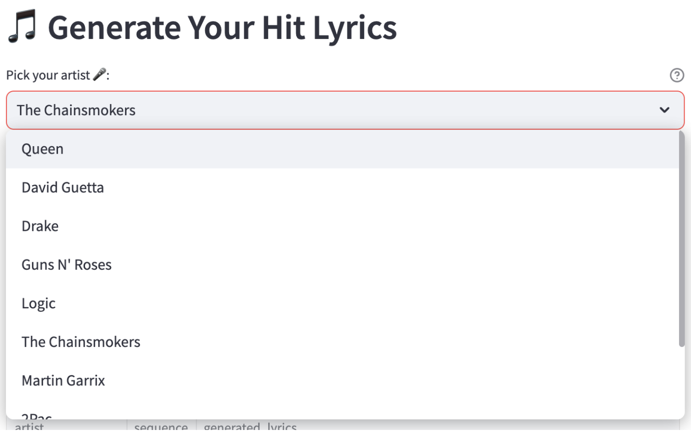
4.2. Artist-Specific Lyrics Generation

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



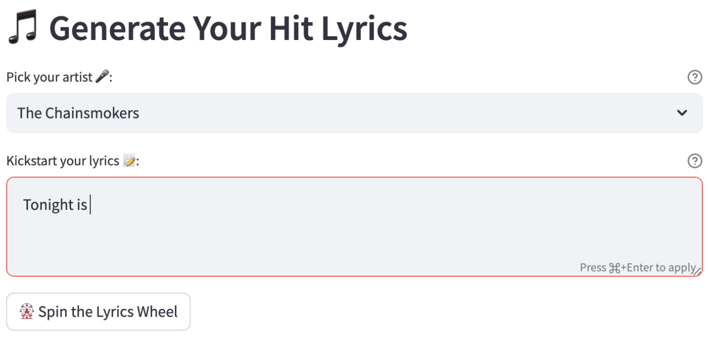
Generate Your Hit Lyrics

Pick your artist 🎤:

The Chainsmokers

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

artist | sequence | generated_lyrics



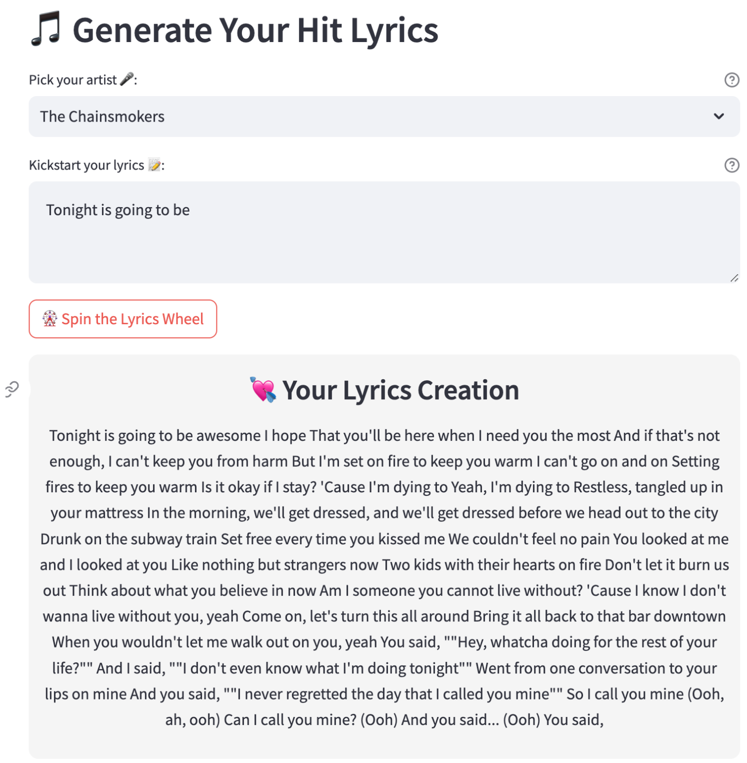
Generate Your Hit Lyrics

Pick your artist 🎤:

The Chainsmokers

Kickstart your lyrics 📝:

Tonight is


🎵 Generate Your Hit Lyrics

Pick your artist 🎤:

The Chainsmokers

Kickstart your lyrics 📝:

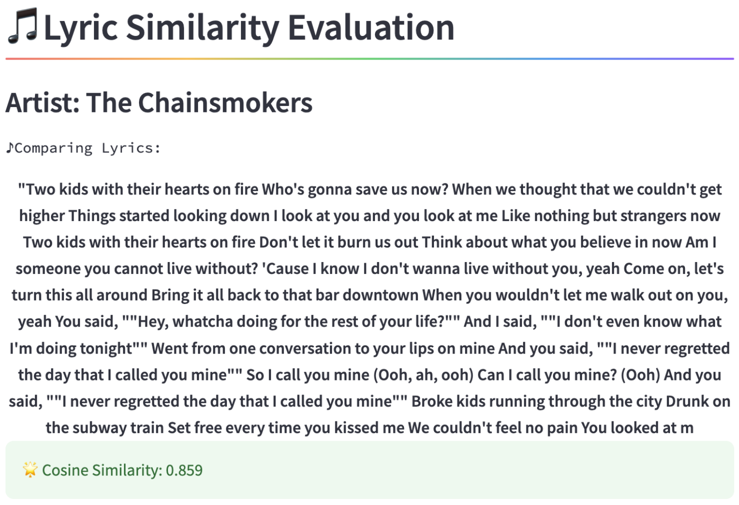
Tonight is going to be

💘 Your Lyrics Creation

Tonight is going to be awesome I hope That you'll be here when I need you the most And if that's not enough, I can't keep you from harm But I'm set on fire to keep you warm I can't go on and on Setting fires to keep you warm Is it okay if I stay? 'Cause I'm dying to Yeah, I'm dying to Restless, tangled up in your mattress In the morning, we'll get dressed, and we'll get dressed before we head out to the city Drunk on the subway train Set free every time you kissed me We couldn't feel no pain You looked at me and I looked at you Like nothing but strangers now Two kids with their hearts on fire Don't let it burn us out Think about what you believe in now Am I someone you cannot live without? 'Cause I know I don't wanna live without you, yeah Come on, let's turn this all around Bring it all back to that bar downtown When you wouldn't let me walk out on you, yeah You said, ""Hey, whatcha doing for the rest of your life?"" And I said, ""I don't even know what I'm doing tonight"" Went from one conversation to your lips on mine And you said, ""I never regretted the day that I called you mine"" So I call you mine (Ooh, ah, ooh) Can I call you mine? (Ooh) And you said... (Ooh) You said,

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.


🎵Lyric Similarity Evaluation

Artist: The Chainsmokers

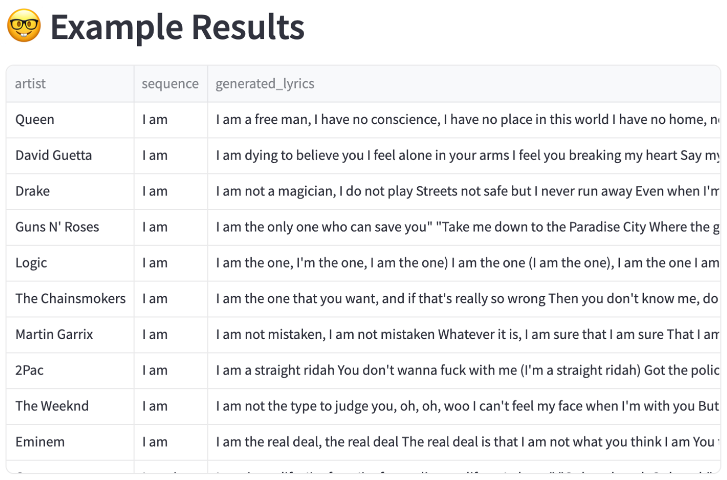
♪Comparing Lyrics:

"Two kids with their hearts on fire Who's gonna save us now? When we thought that we couldn't get higher Things started looking down I look at you and you look at me Like nothing but strangers now Two kids with their hearts on fire Don't let it burn us out Think about what you believe in now Am I someone you cannot live without? 'Cause I know I don't wanna live without you, yeah Come on, let's turn this all around Bring it all back to that bar downtown When you wouldn't let me walk out on you, yeah You said, ""Hey, whatcha doing for the rest of your life?"" And I said, ""I don't even know what I'm doing tonight"" Went from one conversation to your lips on mine And you said, ""I never regretted the day that I called you mine"" So I call you mine (Ooh, ah, ooh) Can I call you mine? (Ooh) And you said, ""I never regretted the day that I called you mine"" Broke kids running through the city Drunk on the subway train Set free every time you kissed me We couldn't feel no pain You looked at m

🌟 Cosine Similarity: 0.859

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.

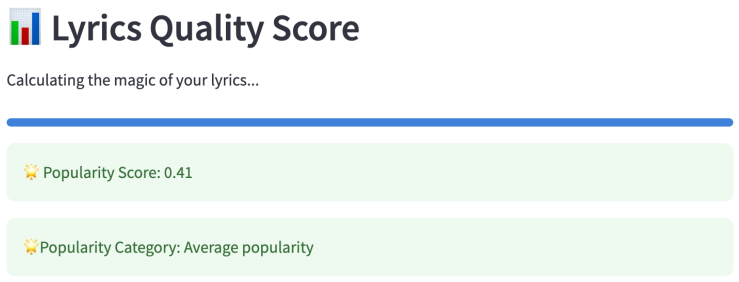


🤓 Example Results

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'r
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 an
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
[illegible]	[illegible]	[illegible]

4.5. Lyrics Quality Score

The app evaluates the quality of the generated lyrics by providing a popularity score and category based on a trained XGBoost model and a regression model.



📊 Lyrics Quality Score

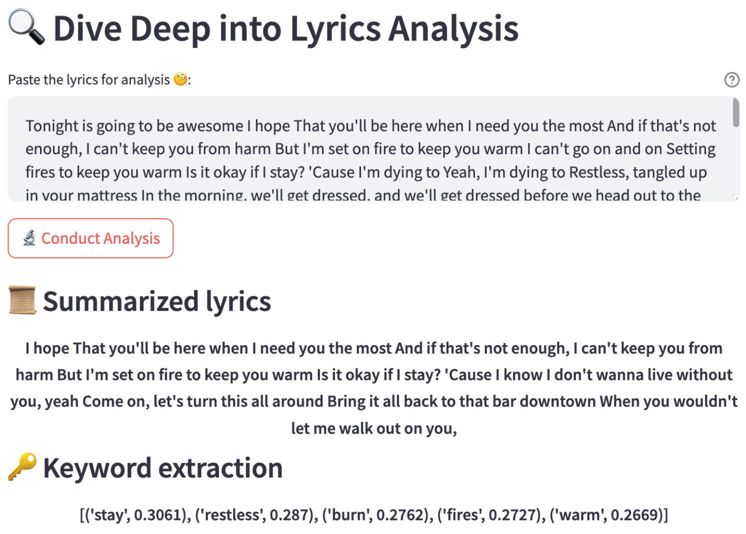
Calculating the magic of your lyrics...

🌟 Popularity Score: 0.41

🌟Popularity Category: Average popularity

4.6. Lyrics Analysis

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


🔍 Dive Deep into Lyrics Analysis

Paste the lyrics for analysis 🧐:

Tonight is going to be awesome I hope That you'll be here when I need you the most And if that's not enough, I can't keep you from harm But I'm set on fire to keep you warm I can't go on and on Setting fires to keep you warm Is it okay if I stay? 'Cause I'm dying to Yeah, I'm dying to Restless, tangled up in your mattress In the morning, we'll get dressed, and we'll get dressed before we head out to the

📜 Summarized lyrics

I hope That you'll be here when I need you the most And if that's not enough, I can't keep you from harm But I'm set on fire to keep you warm Is it okay if I stay? 'Cause I know I don't wanna live without you, yeah Come on, let's turn this all around Bring it all back to that bar downtown When you wouldn't let me walk out on you,

🔑 Keyword extraction

```
[('stay', 0.3061), ('restless', 0.287), ('burn', 0.2762), ('fires', 0.2727), ('warm', 0.2669)]
```

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. 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