Genre-Based Song Lyrics Generation

Gharam Walid, Yousef Hassan

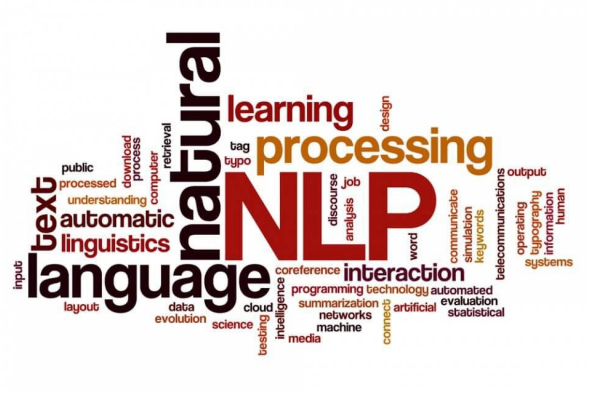
May 19, 2024

1 Introduction

In the era of digital streaming services and personalized playlists, the demand for innovative approaches to music generation and recommendation has surged. Among these, genre-based song lyrics generation stands out as a fascinating avenue, offering a blend of creativity, artificial intelligence, and linguistic analysis. With the advent of advanced machine learning techniques and the abundance of digital music data, researchers and enthusiasts alike are exploring novel methods to automatically generate song lyrics tailored to specific genres. This paper delves into the burgeoning field of genre-based song lyrics generation, discussing its significance, challenges, and recent advancements. By leveraging natural language processing (NLP) algorithms and large-scale datasets, genre-based song lyrics generation holds the potential to revolutionize music creation, empower artists, and enhance listener experiences. Through an exploration of existing methodologies and emerging trends, this paper aims to shed light on the evolving landscape of AI-driven music generation and its implications for the future of music composition and consumption.

2 Literature Review

Neural network-driven language models have demonstrated potential in generating original lengthy written content using minimal initial text, with recent efforts extending to the realm of musical lyric creation. However, crafting lyrics presents a distinct set of challenges not encountered in typical prose. Ensuring appropriate line breaks, mastering stylistic elements such as flow, rhyming, and repetition, and adhering to established lyrical structures like the verse-chorus format are all crucial factors in producing compelling songs. Despite varying degrees of success in these endeavors, it's important to recognize the broad spectrum encompassed by the term "lyrics." Different music genres exhibit unique lyrical styles, characterized by variations in line length, word repetition, semantic nuances, and narrative perspectives. Our research endeavors to explore the capacity of neural networks to generate genre-specific song lyrics while preserving the intrinsic linguistic features characteristic of each musical genre.



LSTM-based lyrical generation has been investigated within defined music genres. In their work "GhostWriter: Using an LSTM for Automatic Rap Lyric Generation," Potash et al. (2015) [Pot15] endeavored to produce lyrics akin to those of a "ghostwriter," crafting content tailored to a particular artist. Nonetheless, their model's scope was constrained to generating lyrics within a singular genre due to its training on a specific artist. Moreover, Potash et al. implemented rhyme schemes by training their model on datasets containing rhyming word pairs.

In [LFM19], the authors introduce iComposer, a system designed to simplify music production by automatically generating melodies and lyrics based on text input. The system is built on a sequence-to-sequence model architecture, with a focus on predicting melody, rhythm, and lyrics. Specifically, the authors utilize LSTM (long short-term memory) networks in their model, which are well-suited for sequence-based tasks due to their ability to capture long-range dependencies in data.

By leveraging LSTM networks, iComposer effectively learns the structure of Chinese popular music and generates melodies and lyrics that align harmoniously. The experimental results demonstrate that iComposer can compose compelling songs at a level comparable to human composers, showcasing the effectiveness of LSTM in capturing the intricate relationships between lyrics and melody. Overall, the integration of LSTM in iComposer enables the system to create music that resonates with listeners, highlighting the potential of LSTM in enhancing automated music composition systems.

In [GLM20] , the paper "Deep Learning in Musical Lyric Generation" explores the application of neural network-based language models, specifically LSTM, in generating original song lyrics across different genres. The study focuses on key linguistic characteristics such as average line length, word variation, point-of-view analysis, and word repetition to capture the essence of various music genres. By training models on specific genres like jazz, country, metal, pop, rap, and rock, the researchers generated lyrics based on 16-word prompts, showcasing the ability of the models to mimic genre-specific lyrical styles.
Despite challenges in capturing certain metrics like point-of-view in rock lyrics, the models demonstrated a high degree of semantic relevance and stylistic accuracy. The research not only highlights the potential of deep learning in music composition but also opens avenues for personalized text generation tailored to specific audience preferences within distinct writing styles.

The following figure is from [GLM20] which shows the percentage changes from original to generated lyrics for each metric and genre. This figure provides a detailed comparison of how the generated lyrics differ from the original ones in terms of various linguistic metrics, offering insights into the model's performance and the extent to which it captures the nuances of different music genres.

Genre	*Average Line Length*	*Song Word Variation*	*Genre Word Variation*	*I vs. You*	*Word Repetition*
Metal	-54.3	-25.1	-13.5	82.7	88.2
Rock	-38.4	-5.0	-5.9	77.0	34.9
Rap	-52.3	-64.6	-99.0	75.5	-49.6
Pop	-28.0	-16.7	-44.6	94.6	36.5
Country	-74.1	-36.6	-23.0	-229.6	75.2
Jazz	-24.1	-38.7	-78.5	73.7	94.6

Figure 1: Percentage Changes from original to generated lyrics for each metric and genre

In [FS14] , the paper presents a novel approach for analyzing and classifying song lyrics. The authors experiment with n-gram models and more sophisticated features to model different dimensions of song texts, such as vocabulary, style, semantics, orientation towards the world, and song structure. They demonstrate the effectiveness of these features in three classification tasks: genre detection, distinguishing the best and worst songs, and determining the approximate publication time of a song. The study draws on earlier work on text classification, poetry analysis, and lyrics-based music classification.
The authors collected their own dataset of English song lyrics and used metadata such as genre information, quality ratings, and publication years for their experiments. They designed 13 feature classes grouped into five abstract sets, and conducted three experiments to classify songs by genre, quality, and publication time. The results show that lyrics-based statistical models can be employed to perform different music classification tasks, and the extended feature set consistently improves classification performance. The study provides insights into the potential benefits of lyrics-based music classification for music retrieval, recommendation systems, and musicology research.

In [GSB], the paper investigates the effectiveness of different RNN-based language modeling architectures for music lyrics classification and generation. The custom dataset contains 80k song lyrics of the genres Rock, Pop, Rap, and Country. The classification performance of Vanilla RNN, GRU, and various LSTM variations is evaluated, with the GRU achieving the best performance. For lyrics generation, LSTM consistently produced higher quality lyrics. The paper also explores the use of initial cell-state and hidden-state to generate lyrics in specific genres. The results show that LSTM and GRU architectures are well-suited for classification and generation tasks. Additionally, the paper discusses the challenges in genre classification and the potential for future work using transformer-based models.

In [SKAO22] the paper explores the application of Bidirectional Long Short-Term Memory (BLSTM) networks for sentiment classification of Covid-19 related online articles. The study evaluates how changing various parameter values impacts the performance of LSTM and BLSTM models. The dataset

consists of over 10,000 articles sourced from prominent news websites, including CNN and Fox News, covering the period from May 2020 to September 2020. The research highlights the significance of parameter tuning in achieving optimal model performance, with experimental results demonstrating that the BLSTM model generally outperforms the unidirectional LSTM in terms of classification accuracy. The findings underscore the robustness of BLSTM models in handling textual data with complex linguistic structures and varying sentiment polarities, providing valuable insights for sentiment analysis tasks in NLP.

In [Kab21], the author proposes a novel sentiment analysis model combining convolutional neural networks (CNN) and bidirectional long short-term memory (BiLSTM) networks for analyzing Turkish tweets about COVID-19. The study emphasizes the inclusion of emojis and emoticons in sentiment analysis and employs a lexicon-based tweet annotation approach. The model, trained on 15,000 Turkish tweets, achieved an impressive accuracy of 97.895%, outperforming state-of-the-art baseline models.This paper's approach to combining CNN and BiLSTM architectures and its focus on handling non-textual elements like emojis provide valuable insights for our project. The high accuracy achieved by this hybrid model underscores the potential benefits of integrating different neural network architectures for complex text analysis tasks.

In the master's thesis by Fredrik Pettersson [Pet19], the focus is on optimizing deep neural networks for the classification of short texts. The research explores how state-of-the-art neural network models can be adapted for specific natural language processing (NLP) datasets and investigates the impact of different dimensionality reduction techniques on commonly used pre-trained word embeddings. The thesis employs convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and bidirectional LSTMs (Bi-LSTMs) combined with various word embeddings like GoogleNews, GloVe, Paragram, and Wiki-news. The research involves three main experiments: creating a top-performing text classification model for a Kaggle competition, reducing the dimensionality of word embeddings using principal component analysis (PCA) and random projection (RP), and benchmarking the model on a secondary dataset (Sentiment140) to evaluate generalization. The findings show that PCA can reduce embedding dimensionality by 66% with minimal accuracy loss, improving execution time by 13%. The Bi-LSTM model achieved an F1 score of 71% in the competition and 86% accuracy on the Sentiment140 dataset, highlighting its generalizability and efficiency.

Finally , In [Mah20] , the author explores an advanced approach to text summarization using deep learning techniques. The study focuses on implementing a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) units, enhanced by an attention mechanism, to generate abstractive summaries of text documents. This method aims to produce summaries that not only capture the essential information but also maintain the semantic and grammatical integrity of the original text. The research involves training the model on a large dataset and evaluating its performance using the ROUGE metric, demonstrating the effectiveness of attention-based mechanisms in improving the quality of abstractive summaries. The findings indicate that incorporating attention significantly enhances the model's ability to understand and summarize long documents, making this approach a valuable contribution to the field of natural language processing.

3 Dataset

In this dataset manipulation process, we first loaded a CSV dataset file using the pandas library in Python. We attempted to read the dataset into a DataFrame, ensuring proper encoding and handling potential parsing errors. Subsequently, we displayed the first few rows of the dataset and retrieved information about its structure, including data types and missing values. After handling missing values and removing duplicate rows, we proceeded to tokenize the text data using the NLTK library, breaking down each artist's name into individual words. Next, we cleaned the tokenized text by converting words to lowercase and removing punctuation and stopwords. Finally, we saved the modified DataFrame to a new CSV file and provided a link for downloading the processed dataset. This comprehensive process allowed us to prepare the dataset for further analysis or modeling tasks.

3.1 Data Analysis

In the given dataset, the analysis focused on determining the most frequent word present in the 'artist' column. After processing the text data by joining the 'artist' column entries into a single string and then splitting them into individual words, a dictionary was created to store the frequency count of each word. Through iteration over the words, their frequencies were tallied in the dictionary. Subsequently, the word with the highest frequency was identified using the max() function, which returned the word 'John'. The frequency of occurrence of this word was found to be 968 times within the 'artist' column. This outcome suggests that 'John' appears most frequently among all the artist names recorded in the dataset. Such analysis provides valuable insights into the distribution of artist names and the prominence of specific terms within the dataset.

The analysis focused on determining the most frequent word present in the 'song' column. The text data from the 'song' column was combined into a single string, and then split into individual words. A dictionary was created to store the frequency count of each word. Iterating over the words, their frequencies were tallied in the dictionary. Subsequently, the word with the highest frequency was identified using the max() function. In this case, the word 'The' was found to be the most frequent, appearing 7933 times within the 'song' column. This outcome suggests that 'The' is the most commonly occurring word among all the song titles recorded in the dataset. Such analysis provides valuable insights into the distribution of song titles and the prominence of specific terms within the dataset.

In this analysis of the dataset, the objective was to identify the most frequent word in the 'text' column. Initially, the textual data from the 'text' column was concatenated into a single string. This string was then split into individual words. A dictionary was employed to track the frequency count of each word encountered. Through iteration over the words, their frequencies were recorded in the dictionary. Subsequently, utilizing the max() function, the word with the highest frequency was determined. In this instance, the word 'the' emerged as the most frequently occurring word in the 'text' column, with a frequency count of 446,872. This finding underscores the prevalence of the word 'the' within the textual content of the dataset, offering valuable insights into common linguistic patterns present in the dataset.

We aimed to identify the largest elements within each column of our dataset. By iterating through the columns, the code computed the maximum length of each element within a column and stored the largest element along with its size. Upon execution, the output revealed the largest elements along with their corresponding sizes for each column in the dataset. For instance, in the 'artist' column, the largest element identified was 'Joseph And The Amazing Technicolor Dreamcoat', with a size of 44 characters. Similarly, the 'song' column featured the largest element 'A Simple Desultory Philippic (Or How I Was Robert McNamara'd Into Submission)' with a size of 77 characters. Notably, the 'text' column exhibited the largest element, a lengthy lyrical excerpt, spanning 3997 characters. These findings underscore the considerable variability in element sizes across different columns, reflecting the diverse nature of the textual data contained within the dataset.

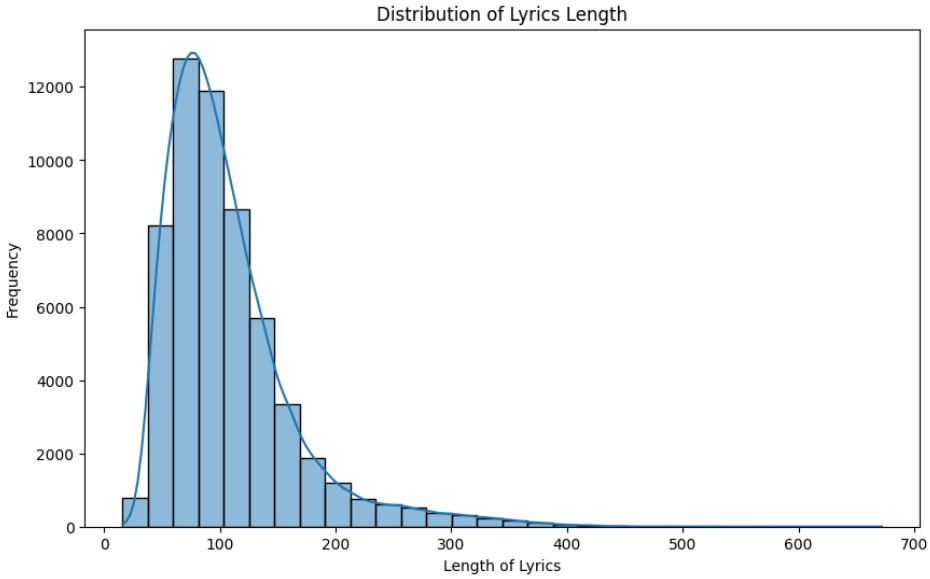


Figure 2: Distribution of Lyrics Length

3.1.1 Distribution of Lyrics Length

The histogram above shows the distribution of the length of lyrics in the dataset. The x-axis represents the length of the lyrics (number of words), and the y-axis represents the frequency (number of songs). We observe that the majority of the lyrics fall within the range of 50 to 150 words. This indicates that most songs in the dataset have relatively short lyrics, with a decreasing number of songs as the lyrics length increases. This distribution helps us understand the typical length of song lyrics and can guide the preprocessing steps in terms of padding and truncation during model training.

3.1.2 Top 10 Most Common Artists

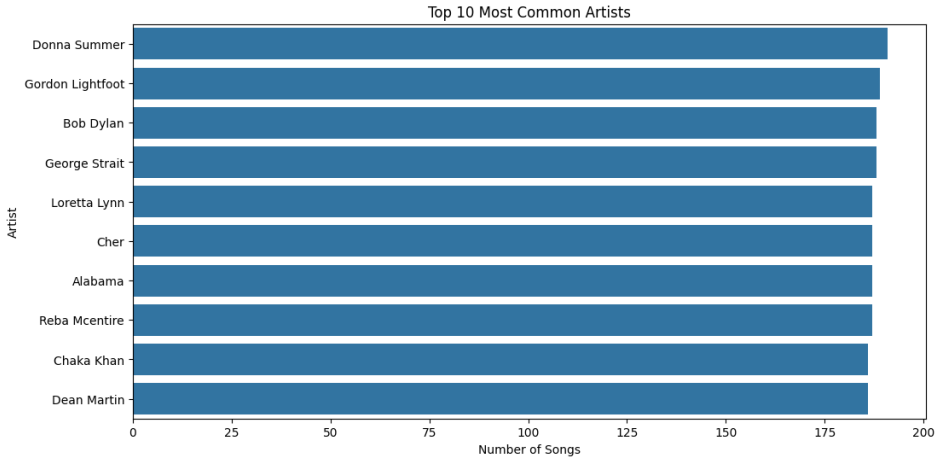


Figure 3: Top 10 Most Common Artists

The bar chart above shows the top 10 most common artists in the dataset, based on the number of songs attributed to each artist. Donna Summer, Gordon Lightfoot, and Bob Dylan are among the most frequently occurring artists, each with nearly 200 songs. This analysis provides insight into the distribution of songs across different artists and highlights the prevalence of certain artists in the dataset. Understanding which artists are most represented can help in tailoring genre-specific models or focusing on artist-specific lyric generation tasks.

3.1.3 Top 20 Words by TF-IDF Score

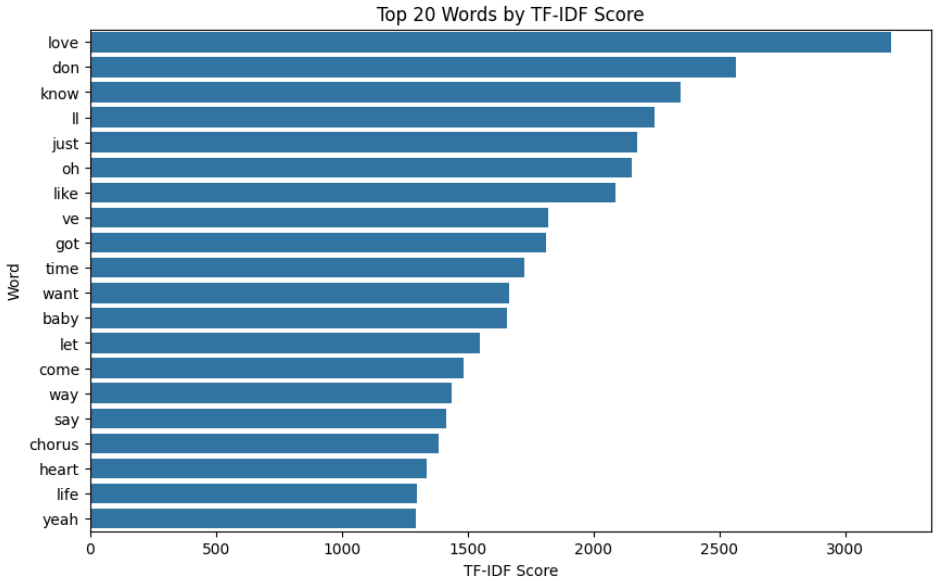


Figure 4: Top 20 Words by TF-IDF Score

The bar chart above illustrates the top 20 words in the lyrics dataset based on their TF-IDF scores. TF-IDF (Term Frequency-Inverse Document Frequency) is a metric used to evaluate the importance of a word in a document relative to a collection of documents (corpus). Words like "love", "don", "know", and "ll" have the highest TF-IDF scores, indicating their significance in the lyrics. This analysis helps identify the most impactful words in the dataset, which can be crucial for understanding the thematic elements of the songs.

3.1.4 Top 20 Most Common Words

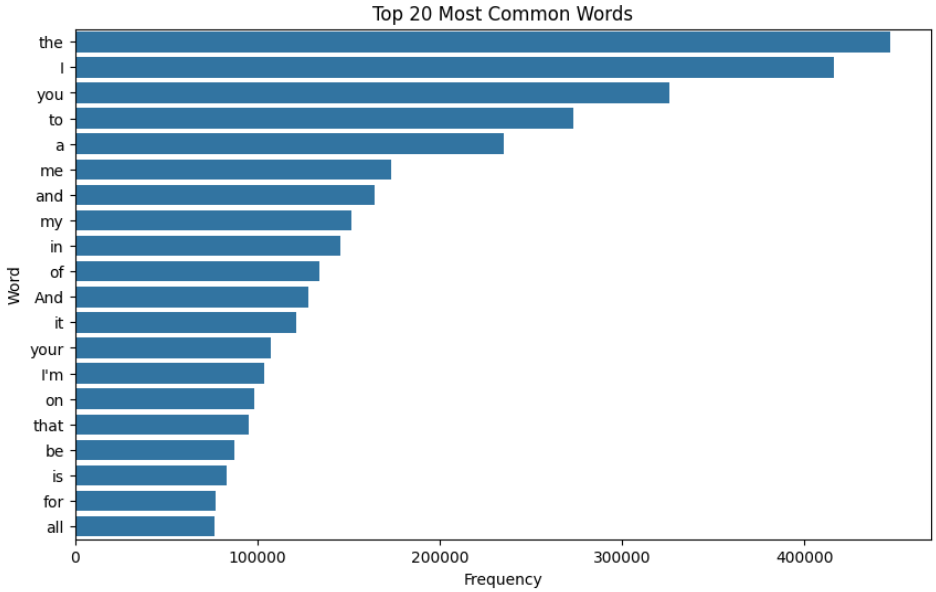


Figure 5: Top 20 Most Common Words

The bar chart above shows the top 20 most common words in the lyrics dataset. The x-axis

represents the frequency of each word, while the y-axis lists the words. Common stopwords like "the", "I", "you", and "to" appear most frequently, which is typical in text datasets. Despite their high frequency, these words often carry less semantic weight in the context of the lyrics, as they are used for grammatical purposes. Understanding the frequency of common words helps in preprocessing steps such as stopword removal for better model performance.

3.1.5 Distribution of Words per Document

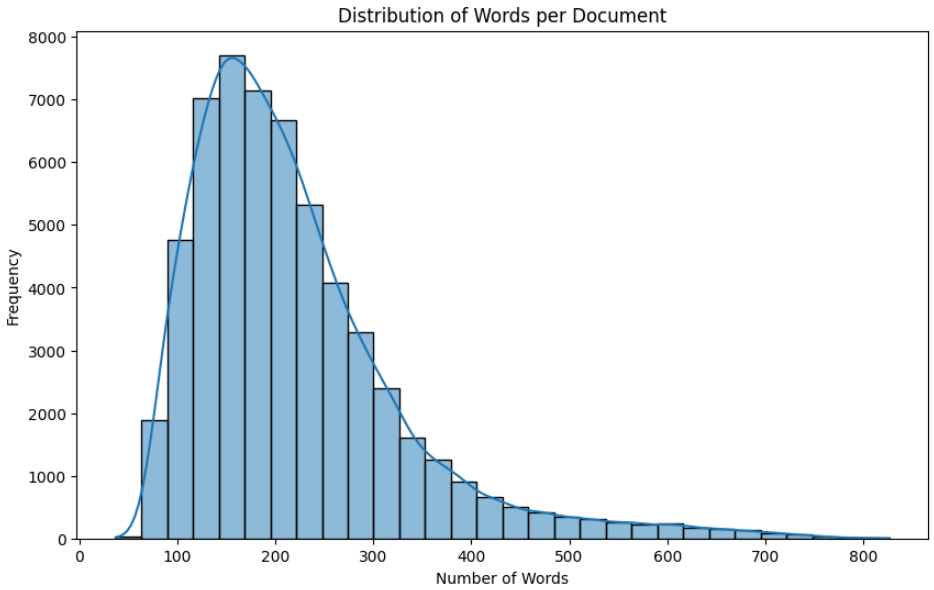


Figure 6: Distribution of Words per Document

The histogram above displays the distribution of the number of words per document (song) in the dataset. The x-axis represents the number of words, and the y-axis represents the frequency of documents with that word count. Most documents have between 100 to 300 words, with a peak around 200 words. This distribution helps in setting the sequence length for models, ensuring that the majority of documents are appropriately represented without excessive padding or truncation.

3.2 Dataset Limitations

In spite of the richness of the dataset, several limitations warrant consideration. One notable limitation is the absence of null values or duplicate entries within the dataset. While the absence of null values and duplicates ensures data integrity and reliability, it also contributes to the dataset's substantial size. As a consequence, managing and processing such a voluminous dataset may pose challenges in terms of computational resources and time efficiency.

We obtained an empty dataframe after performing correlation analysis, it typically means that there is no linear relationship between the variables in the dataset.

It's essential to acknowledge another limitation observed within the dataset: the presence of identical song names attributed to different artists. This phenomenon introduces ambiguity and potential confusion, as song titles may lack the specificity required to uniquely identify a musical composition.

4 Methodology

4.1 Milestone 2: Building a Neural Network Model

In this milestone, our objective was to design and implement a neural network model entirely from scratch to generate song lyrics using a specific dataset. This section provides a comprehensive overview of the methodology we employed, delves into the intricate architecture of the neural network, explains the training regimen, and presents the evaluation outcomes. Our focus was to understand the fundamental aspects of neural network design and how they can be applied to natural language processing tasks without relying on pre-trained models.

The methodology adopted for this project encompassed several critical steps, each essential to ensure the robustness and efficacy of the neural network model:

4.1.1 Data Loading and Preprocessing

- We started by importing the dataset, which comprised song lyrics from various artists. Using the pandas library, we read the dataset into a DataFrame for easy manipulation and analysis. This dataset was analyzed earlier, showing a distribution of lyrics lengths and identifying the most common artists.
- To maintain consistency and focus, we identified the artist with the highest number of songs in the dataset, as shown in our previous analysis where Donna Summer was the most frequent artist. This step involved using the `value_counts` method to count the number of songs per artist and then filtering the dataset to include only the songs of the most prolific artist. This filtering process ensured that the training data was homogeneous in terms of style and language.
- Tokenization was performed using the `Tokenizer` class from TensorFlow Keras, which converts the text data into sequences of integers. Each word in the lyrics was assigned a unique integer, allowing the model to process the text in a numerical format. The earlier analysis of the most common words and their TF-IDF scores provided insights into the vocabulary used in the lyrics.
- We created input sequences and their corresponding target words. For each lyric, we generated n-gram sequences where the input was a sequence of words and the target was the next word in the sequence. This setup is crucial for training the model to predict subsequent words based on preceding context.
- Padding was applied to ensure that all input sequences had the same length. This involved adding zeros to the beginning of shorter sequences, making them equal in length to the longest sequence in the dataset. Consistent input lengths are necessary for the neural network to process the data effectively. The distribution of words per document, analyzed earlier, helped determine appropriate sequence lengths.

4.1.2 Model Building

- The architecture of the model was designed using the Sequential API from TensorFlow Keras. This straightforward approach allowed us to stack layers in a linear fashion, defining the flow of data through the network.
- The first layer was an Embedding layer, which converts the integer-encoded words into dense vectors of fixed size. This layer is essential for capturing the semantic meaning of words and reducing the dimensionality of the input space. The earlier analysis of TF-IDF scores helped in understanding the importance of embedding layers to capture word meanings effectively.
- Two Long Short-Term Memory (LSTM) layers were added next. LSTMs are a type of recurrent neural network (RNN) capable of learning long-term dependencies in sequential data. The first LSTM layer was configured to return sequences, allowing the subsequent LSTM layer to process the entire sequence of hidden states.

- A Dropout layer was incorporated between the LSTM layers to mitigate overfitting. Dropout works by randomly setting a fraction of the input units to zero during training, which helps the network generalize better to unseen data.
- The final layer was a Dense output layer with a softmax activation function. This layer provided a probability distribution over the vocabulary, enabling the model to predict the likelihood of each word being the next word in the sequence.

4.1.3 Training Process

- **Data Splitting:** The dataset was divided into training and testing sets to allow for model validation and evaluation. The training set comprised 90% of the data, while the testing set comprised 10%. This split ensured that the model could be evaluated on a separate dataset, providing a realistic measure of its generalization performance.
- **Callbacks:**
  - **Early Stopping:** Early stopping monitored the validation loss and halted training if the loss did not improve for a specified number of epochs (patience). This helped prevent overfitting by stopping training when the model was no longer improving.
  - **Model Checkpointing:** Model checkpointing saved the best model based on validation loss during training. This ensured that the best-performing model was retained, even if subsequent epochs resulted in worse performance.
  - **Learning Rate Reduction:** Learning rate reduction dynamically adjusted the learning rate when the validation loss plateaued. Lowering the learning rate helped fine-tune the model and led to better convergence.
- **Training:** The model was trained for five epochs with a batch size of 128. During each epoch, the model learned to predict the next word in a sequence by minimizing the prediction error. The validation set was used to monitor the model's performance and apply the configured callbacks. The use of early stopping and model checkpointing ensured that the best model was saved, and the training process was halted once the model's performance no longer improved.

4.1.4 Model Evaluation and Text Generation

After training the model, we implemented a function to generate text based on a given seed text. This function demonstrates the model's practical application in generating song lyrics by predicting the next word in a sequence. The function operates as follows:

- **Initialization:** The function starts with a seed text, which serves as the initial context for generating new words. This seed text is crucial as it provides the initial context for the model to begin making predictions.
- **Tokenization and Padding:** The seed text is first tokenized into a sequence of integers using the trained tokenizer. These integers correspond to the words in the tokenizer's vocabulary. Since neural networks require fixed-length input, the tokenized sequence is padded to ensure it matches the maximum sequence length expected by the model. Padding involves adding zeros to the beginning of the sequence if it is shorter than the maximum length.
- **Prediction:** The padded sequence is then fed into the trained model, which generates a probability distribution over the entire vocabulary. This distribution indicates the likelihood of each word in the vocabulary being the next word in the sequence. The model uses its learned patterns to make this prediction.
- **Selecting the Next Word:** The word with the highest probability is selected as the predicted next word. This is done by identifying the index with the highest probability value from the model's output. The tokenizer's index-to-word mapping is used to convert this index back into the corresponding word.

- **Updating the Text:** The predicted word is appended to the current text, extending the sequence. The function then repeats the process: tokenizing the updated text, padding it, and predicting the next word. This iterative process continues until the desired number of new words has been generated.
- **Handling Invalid Predictions:** If the predicted word is invalid (e.g., it is `None` or an empty string), the function skips it and continues with the next iteration. This ensures that only meaningful words are added to the generated text.
- **Final Output:** Once the specified number of words has been added, the function returns the complete text, which now includes the seed text followed by the newly generated words. This method effectively demonstrates the model's ability to generate coherent and contextually appropriate lyrics by leveraging its learned knowledge.

After completing the training process, the model was evaluated using a test dataset to measure its performance. The evaluation metrics obtained were:

- **Test Loss:** 5.2321
- **Test Accuracy:** 0.1230

Interpretation of Test Loss and Accuracy

The test loss is a measure of how well the model's predictions match the true labels on the test dataset. A lower loss indicates a better fit of the model to the data. In this case, the test loss is 5.2321, which is relatively high. This suggests that the model has not achieved a good fit and that there may be significant errors in its predictions. Several factors could contribute to this high test loss:

- Model Complexity: The model might be too complex for the given data, leading to overfitting during training but poor generalization to unseen data.
- Data Quality: The quality and diversity of the training data play a crucial role. If the training data does not adequately represent the test data, the model may struggle to perform well.
- Training Duration: The model might require more epochs to learn from the data effectively. Conversely, it might also indicate that the model has been overfitted, necessitating early stopping or regularization techniques.

Test accuracy measures the proportion of correct predictions made by the model on the test dataset. An accuracy of 0.1230 means that the model correctly predicts only 12.30% of the time, which is significantly below the expected performance for a robust model.Several reasons could explain this low accuracy:

- Imbalanced Dataset: If the dataset is imbalanced (e.g., more negative samples than positive), the model might struggle to learn the minority class, leading to poor performance.
- Inadequate Training: The model might not have been trained for a sufficient number of epochs, or the learning rate might be too high or too low, preventing the model from converging to an optimal solution.
- Model Architecture: The chosen model architecture might not be suitable for the complexity of the task. Adjustments to the architecture, such as adding more layers, changing activation functions, or using different types of layers (e.g., combining CNN with LSTM), could improve performance.

The implemented text generation function highlights the model's practical application in generating song lyrics. By predicting and appending words iteratively, the function showcases the model's

capability to create text that maintains the style and structure of the training data. This function underscores the potential of neural networks in natural language processing tasks, particularly in creative text generation.

This comprehensive approach, integrating data analysis and preprocessing, ensured that the model was trained on a well-structured and representative subset of the dataset. The architecture and training strategies were chosen based on the nature of the data, aiming to optimize the model's ability to generate coherent and contextually appropriate song lyrics.

References

[FS14] Michael Fell and Caroline Sporleder. Lyrics-based analysis and classification of music. pages 620–631, 2014.

[GLM20] Harrison Gill, Daniel Lee, and Nick Marwell. Deep learning in musical lyric generation: an lstm-based approach. *The Yale Undergraduate Research Journal*, 1(1):1, 2020.

[GSB] Daniel Gordin, Arnab Sen Sharma, and Jaydeep Borkar. Music lyrics classification & generation using rnns.

[Kab21] Abdullah Talha Kabakus. A novel covid-19 sentiment analysis in turkish based on the combination of cnn and bilstm on twitter. *Department of Computer Engineering, Faculty of Engineering, Duzce University, Duzce, Turkey*, 2021.

[LFM19] Hsin-Pei Lee, Jhih-Sheng Fang, and Wei-Yun Ma. icomposer: An automatic songwriting system for chinese popular music. pages 84–88, 2019.

[Mah20] Rojina Maharjan. Abstractive text summarization using recurrent neural network with attention based mechanism. 2020.

[Pet19] Fredrik O. Pettersson. Optimizing deep neural networks for classification of short texts. 2019.

[Pot15] Romanov A. Rumshisky A. Potash, P. Ghostwriter: Using an lstm for automatic rap lyric generation. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.*, 2015.

[SKAO22] Xiaoyu Li Samuel Kofi Akpatsa, Hang Lei and Victor-Hillary Kofi Setornyo Obeng. Sentiment classification of covid-19 online articles: The impact of changing parameter values on bidirectional lstm. *International Journal of Engineering Research & Technology (IJERT)*, 11(1):526–532, 2022.