NLP Project

Nada Bakeer 49-0360, Nour Shehab 49-0463

May 19, 2024

1 Motivation

This paper investigates the research domains of natural language processing and information retrieval in the context of lyrics production, depending on artist names. By analyzing a vast dataset of lyrics associated with certain artists, we expect to uncover patterns and insights influenced by their styles and personalities. This technique applies NLP and IR research to artistic expression, potentially impacting music recommendation systems and tailored content production.

2 Introduction

In this paper we are going to be discussing some of the previous work, followed by data preparation and analysis to prepare our dataset for the training process.

3 Literature Review

In a paper by Gill et al.[Gom22], a method for generating lyrics using deep learning was explored. They create a word dictionary from lyrics from various musical genres and use a neural network to anticipate the next word, newline, or punctuation in the lyrics. The network design consists of an embedding layer, an LSTM layer with dropout for regularization and a linear layer to generate a probability vector based on the vocabulary. Due to computational constraints, training was limited to six genres. Results showed that a specific input size (k = 16) effectively captured overarching structural dependencies in lyrics, such as verse-chorus structures, whereas smaller values caused complications such as repetition. The study's limitations were constraints in computational resources that restricted genre selection for training and limited the amount of the training datasets. Furthermore, the task of adapting pre-existing models such as GPT-2 from prose to lyrical creation highlighted the complexity of transferring models across different writing styles. Future research suggestions include exploring alternative model types such as Markov Models and GRUs Overall, the study shows potential in creating lyrics for various genres utilizing modern deep learning algorithms.

In another study, Saeed et al.[SIZ19] presented Creative-GAN, a modified Generative Adversarial Network (GAN) designed for the development of creative literature such as poems, lyrics, and metaphors. Their approach involved training a language model using backpropagation over time, with a focus on maximum likelihood estimation (MLE) during the pre-training phases and including discriminator creative rewards during GAN training. The discriminator gives useful feedback to the generative model via a cost function that encourages the use of inventive tokens. Creative-GAN consists of a model trained using the AWD-LSTM and TransformerXL architectures, as well as a discriminator with an encoder similar to the generator encoder but with additional decoding layers. Based on the input sequence's hidden states, the discriminator decoder generates a numerical output between 0 and 1. Creative-GAN was tested on a variety of creative datasets, including classical and contemporary English poems, metaphor sentences, and song lyrics. It outperformed baseline language models and GumbelGAN models in terms of perplexity scores, indicating improved performance in generating creative text. Nonetheless, the non-differentiable nature of text presents issues, as does the previously described GPT-2 technique[Gom22]. To address this, the generator model parameters are changed using policy gradients. Although Creative-GAN produces promising results in creative text production,

additional optimisation of the architecture and training methods may be required for various creative writing jobs.

Rodrigues et al.[RdPOMdAP22]present a novel way for creating song lyrics by refining a pre-trained GPT-2 model with English and Portuguese lyric datasets. Using the multitasking capabilities of the GPT-2 model, which is built on the Transformer architecture and offered by OpenAI, the project intends to develop musically coherent and grammatically sound lyrical content using a model that was not specifically designed for this purpose.

A study of the created lyrics is offered, with emphasis on spelling, grammar, and semantics. The research digs into efforts to identify underlying similarities in the generated texts and compares them to human-authored song lyrics. The findings show promise in using pre-trained models such as GPT-2 to generate poetic content, highlighting the benefits of transfer learning in optimizing computational resources.

However, significant limitations are highlighted, such as potential issues in dealing with metaphorical language which was also highlighted in the previously mentioned papers, the use of synonyms, and paraphrasing in the created lyrics. Furthermore, the study may not fully capture all aspects of lyrical creation and may require additional modification to improve the quality and diversity of the created texts.

In this paper, Madhumani et al.[MYH+20] use automatic neural lyrics and melody composition (AutoNLMC). The AutoNLMC comprises two models, the first being the lyrics generator, which utilizes a recurrent neural network (RNN) that generates the upcoming token. In other words, the RNN contains a hidden state that is used to build the base upon which the model will predict the next token which in this case is a sequence of syllables. The objective of this model is to learn the syllable distribution and to predict it later on. The RNN uses the input tokens to adjust the hidden state, for it to be used in the upcoming token prediction, as the essence of RNN is that it generates the new token based on the previous tokens as well as the current token (input). This is executed using a non-linear function that could be a sigmoid function or long short-term memory (LSTM).
Moving on to the melody composer which operates using an RNN as well. The model uses a sequential encoder-decoder, where the melody composer model takes lyrics tokens as input to process and outputs hidden states. The hidden state is then passed on to the decoder to generate a melody. Lastly, the encoder generates a dynamic context vector that encodes parts that should be emphasized for the melody composer RNN model. The proposed model was able to generate 10 full songs that were constrained to 100 syllables, the length of the seed lyrics used for the model input was variable. Multiple forms of lyrics encoding were used, thus models were trained separately for each representation. The forms used were syllable embedding (SE), addition of syllable and word vector (ASW), syllable and word embedding concatenation (SWC), and concatenated syllable, word, and syllable-projected word vector (CSWP). Conclusively, both embedding forms ASW and CSWP proved to be of better performance in identifying the correlation between lyrics and melodies.

4 Data Preparation and Analysis

4.1 Dataset

The dataset chosen in this project, is the 57,650 spotify songs dataset. It was chosen as it has a wide variety of artists and each artist has multiple song, which helps in not overfitting the model.

4.2 Data Cleaning

In this section we will be discussing the data cleaning processes done to prepare the data for usage. Initial exploratory data analysis involved checking for null values, examining data types, and summarizing the dataset. We renamed the 'text' column to 'lyrics', removed missing values and duplicates, however none were found, and dropped unnecessary columns such as 'link' and 'song'.

The next step was renaming the column "text" to "lyrics" for a clearer more expressive title. The next step was checking for full row duplicates and removing them if any, however the dataset was clean of that too. The cleaning process then involved extensive text preprocessing. We removed non-

alphabetic characters and certain repetitive terms like 'chorus', 'verse', 'intro', 'original', and 'outro'. The lyrics were converted to lowercase, and additional unwanted characters were removed using regular expressions. This ensured the text was in a consistent format suitable for further analysis. The same was done for the artist names.

When preparing the data we decided against stop words removal and lemmatization to avoid changing the sentence structure which is a key part of the song rhyme and over all coherence. Spelling check was also ignored to keep words such as "sleepin'"form changing the rhyme to "sleeping".

By this we conclude the data cleaning and prepare for the data analysis.

4.3 Data Analysis

To be able to study the dataset data analysis is required.

We first visualized the distribution of songs per artist. Using bar plots, we displayed the number of songs by the top 10 and bottom 10 artists, providing insights into the dataset's composition. This helped us understand which artists had the most and least data points in the dataset. This helps in estimating the most songs the model would need to analyze at a time. The results showed that the top artist is "Donna Summer" with 191 songs in the dataset. This can be shown in Figure 1 and Figure 2.

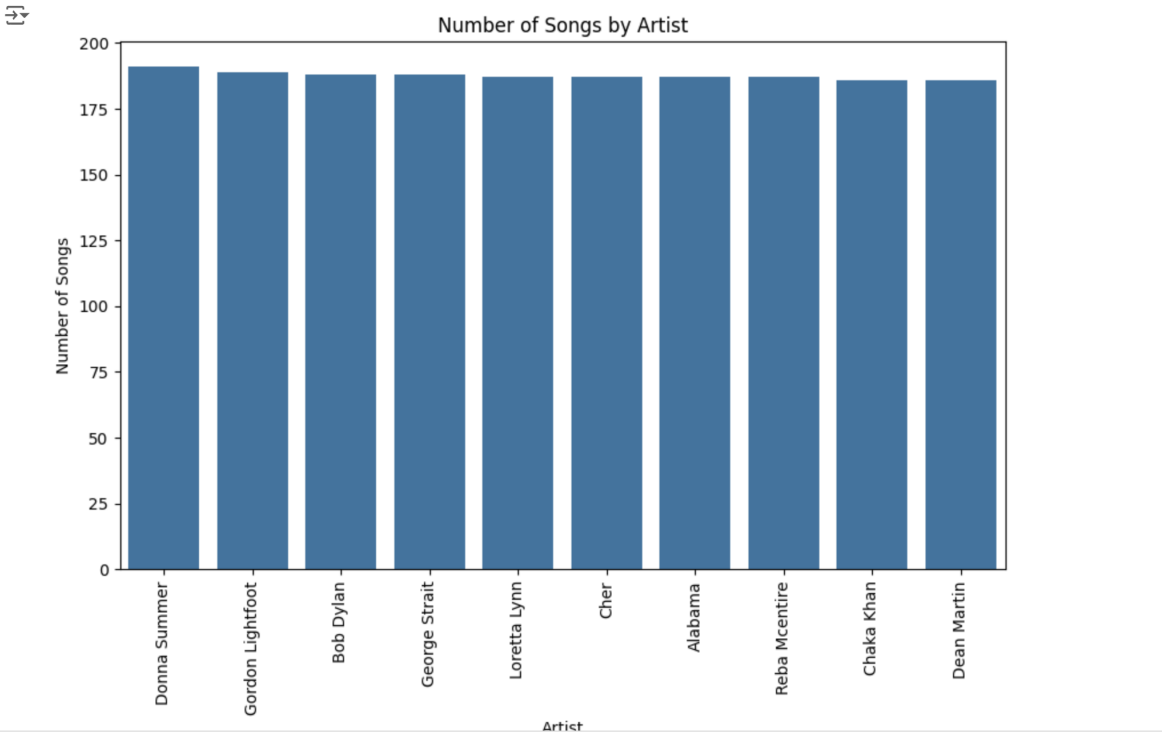

Figure 1: The top most frequent artists number of songs

The second data analysis method we implemented was figuring out the most frequent word per each artist songs. This created a table of artists and the most frequent word they use. This may be helpful in analysing the model output later on.

Using the most frequent word per artist we then calculated the tf-idf for each which might be helpful later on for keyword extraction and information retrieval. This step was crucial for understanding the common themes and words in the lyrics of different artists. To remove stop words and determine the most frequent word for each artist, we developed a function and visualized these frequencies using a bar plot, as shown in Figure 3.

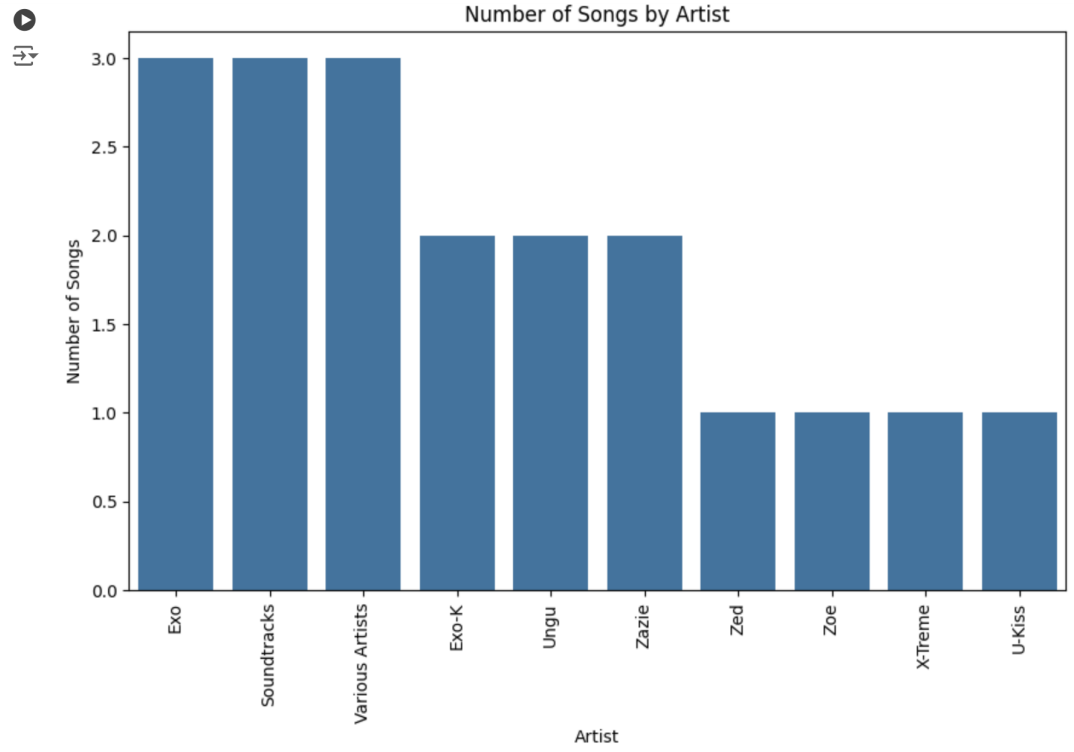

Figure 2: The least frequent artists number of songs

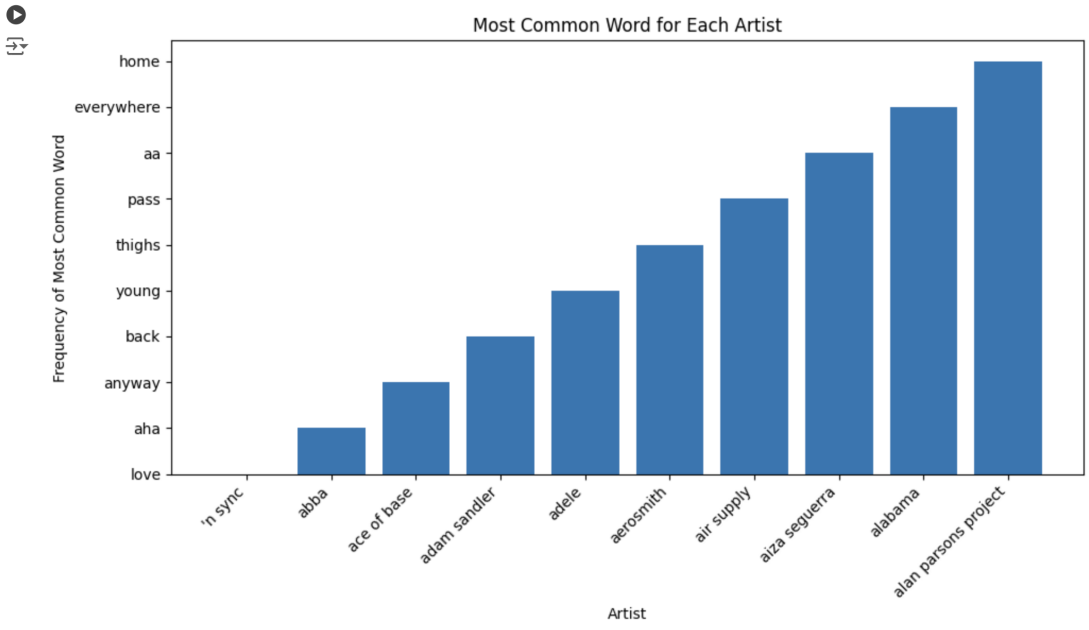

Figure 3: The most frequent word for the first 10 artists

We then analyzed the average song length per artist and the distribution of song lengths. We visualized the average song length for the top 10 artists using bar plots (Figure 4) and created histograms to show the distribution of song lengths across the dataset (Figure 5). This analysis provided further insights into the characteristics of the songs in our dataset. This showed that very few songs are above the length of 1500 words, hence they were truncated.

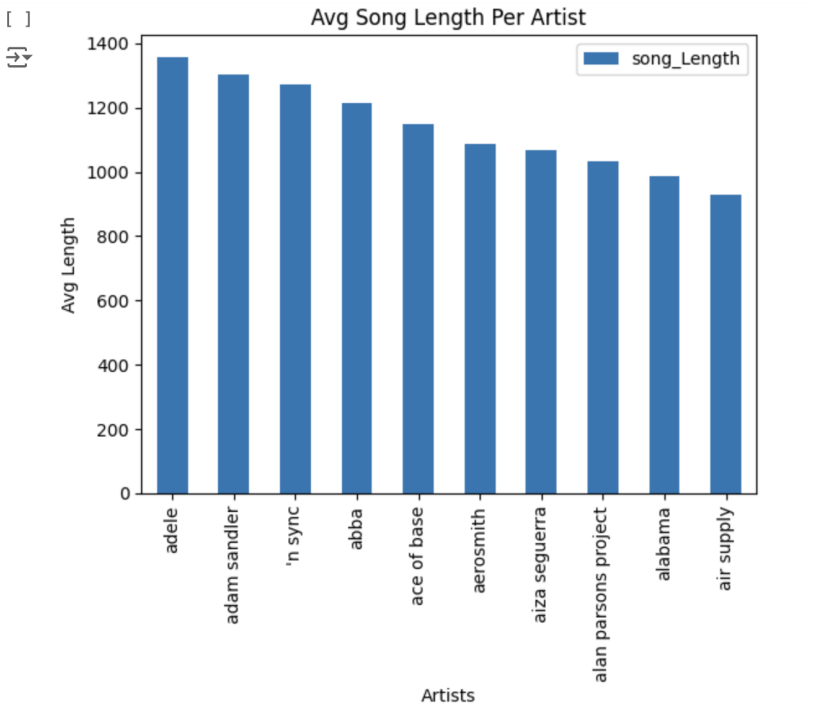

Figure 4: Average Song length for the first 10 artists

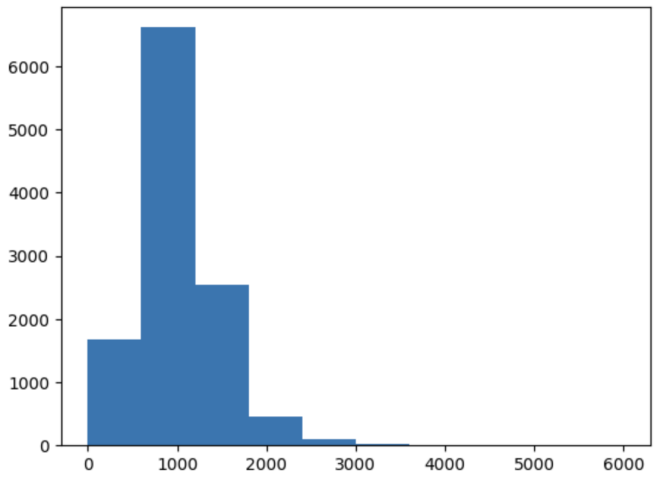

Figure 5: Distribution of song lengths across the dataset

5 MS2: Building the Model

For this milestone, we built a neural network model. We decided to only sample the songs of the first 100 artists in the dataset, as a representation of the whole dataset that had 643 different artists. Then

we joined both the artist name and the lyrics together. The next step was to create a new column containing the last word of each song simultaneously removing this word from the song itself as well. We then removed any song that has more than 1500 in song length.

Moving on, we converted the lyrics data from a DataFrame into a RaggedTensor, which is a nested variable-length list, allowing us to handle the varying song length. We then extract the unique words from these lyrics, providing a sense of the vocabulary size. Later we converted the lyrics to one-hot encoded vectors to be processed by the model, also we indexed the vocabulary present in the dataset.

Next, we split our dataset into training and testing sets with a ratio of 80% and 20% respectively and extract the lyrics and associated labels for both sets. The labels are then one-hot encoded as well. For text tokenization, we use the Tokenizer from Keras, which we configure to filter out unwanted punctuation and convert the text to lowercase. The tokenizer is fitted on the training lyrics data, creating a word index that maps words to unique integers. We then convert the lyrics to sequences of these integers and pad them to ensure uniform input length.

Moreover, with the preprocessed data, we define our neural network model. The model consists of an Embedding layer to convert word indices to dense vectors of a specified dimension, followed by a GRU layer to capture sequential dependencies, a Dense layer with ReLU activation for non-linear transformations, and a final Dense layer with a sigmoid activation function to produce the output probabilities for each word in the vocabulary. We compile the model with the Adam optimizer and categorical cross-entropy loss function. This setup prepares the model to learn from the training data and evaluate its performance on the test data.

Finally, the model was trained for 10 epochs, as for the model evaluation, the resulting loss was 6.977. In addition, we defined a function to map the output probabilities from our model back to words using the vocabulary. This function identifies the word with the highest probability for a given set of probabilities. The main purpose of this function was to compare the actual eliminated word with the model's predicted output.

6 MS3: Using Pre-Trained Model

In this milestone, we trained a Bert generation (BERT2BERT) model to generate the second half of song lyrics given the first half.

Each song's lyrics were split into two halves, the input text was created by combining the artist's name with the first half of the lyrics, with the bert2bert default separation token in between and adding the default start and end token to the text, while the output text was formatted by adding the default bert2bert start and end tokens to the second half of the lyrics. We then removed songs longer than 1500 characters to somewhat standardize the data.

We then split the data into training and test sets. For the model, we loaded the BERT encoder and decoder for the BERT2BERT model and initialized a tokenizer, all using the bert_base_uncased version as the bert_large_uncased version was not supported by our RAM capacity and the bert_tiny_uncased version did not provide the best results. We choose the Bert Tokenizer for easy integration into the pipeline. The training data was then tokenized and converted to PyTorch tensors, and a DataLoader was created for batching the data during training, we used batches of size 8 as larger batches required unavailable RAM capacity. We defined the optimizer and loss function, and trained the model for ten epochs which took around 7 hours, printing the loss for each batch.

To evaluate the model, we generated text for a sample input for a visual assessment and then assessed the model on the test data. This involved processing the test data, the same way the training data was processed, first by tokenizing the test set using the Bert tokenizer, then converting them to PyTorch tensors and finally creating a DataLoader. The evaluation was done by running the model

on the test dataset and calculating the mean loss. the resulting mean loss was approximately (0.02).

In conclusion, BERT2BERT had a much lower loss value comparing to our GRU-based model. There are many reasons for these results, such as BERT being bidirectional, meaning it considers both left and right context when encoding a token, whereas GRUs process sequences in one direction, which can limit their ability to capture context. Also, BERT understands context over long sequences leading to a more complex understanding of context compared to GRUs, which process sequences sequentially. lastly, would be that BERT is pre-trained on large datasets making the model more effective and reliable for tasks like lyrics generation.

References

[Gom22] Walid Gomaa. Lyrics analysis of the arab singer abdel elhalim hafez. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22:1 – 27, 2022.

[MYH+20] Gurunath Reddy Madhumani, Yi Yu, Florian Harscoët, Simon Canales, and Suhua Tang. Automatic neural lyrics and melody composition. *arXiv preprint arXiv:2011.06380*, 2020.

[RdPOMdAP22] Matheus Augusto Gonzaga Rodrigues, Alcione de Paiva Oliveira, Alexandra Moreira, and Maurílio de Araújo Possi. Lyrics generation supported by pre-trained models. *The International FLAIRS Conference Proceedings*, 2022.

[SIZ19] Asir Saeed, Suzana Ilić, and Eva Zangerle. Creative gans for generating poems, lyrics, and metaphors. 09 2019.