

NLP CS 6120: LyricLab

Shikha Tiwari¹, Laasya Anantha Prasad¹, Krishna Venkatesh¹

¹Khoury College of Computer Sciences

¹Northeastern University

Boston, USA

tiwari.shi@northeastern.edu, ananthaprasad.l@northeastern.edu, hosahudyavenkatesh.s@northeastern.edu

Abstract—Lyrics generation necessitates that the model comprehends the context, meaning, and structure of the language. In this study, we compare various NLP models for generating song lyrics: Language N-Gram model, Long Short-Term Memory (LSTM), and Transformer. To assess the models' performance, we train them on Ed Sheeran's song dataset and use metrics such as perplexity and ROUGE score to measure their ability to predict the next word in the sequence. Overall, this study illustrates the potential of NLP techniques in lyric generation and emphasizes the superiority of deep learning models like RNN LSTM and Transformers for this task. The findings could be beneficial for developing lyric generation systems in the music industry, aiding songwriters and artists in creating lyrics more efficiently and effectively.

Keywords—*N Gram, LSTM (Long Short-Term Memory), RNN, GPT2, Transformers, Perplexity, ROUGE Score, BELU Score*

I. INTRODUCTION

Lyrics generation is a significant task in natural language processing (NLP) and has attracted considerable attention due to its potential applications in the music industry. The ability to generate creative and meaningful lyrics can be a valuable tool for songwriters and artists seeking to create new music content. However, generating lyrics is a complex task that requires understanding the language's context, meaning, and structure. Creating song lyrics using a machine learning model has numerous potential applications in the music industry, from aiding songwriters in generating new ideas to providing inspiration for musicians experiencing writer's block. However, producing lyrics that not only sound good but also adhere to standard structure and rhyme patterns is a challenging task for AI models.

In response to these challenges, our project aims to develop a text generation model capable of producing song lyrics that follow the essential components of a song, including an introduction, verse, pre-chorus, chorus, bridge, and outro. We plan to achieve this by employing a combination of NLP techniques, such as N-Gram models and RNN LSTMs, to capture the statistical patterns of language and generate sequences of words that are both grammatically correct and semantically coherent. We also plan to implement a rhyming algorithm to ensure that the end of each sentence or verse in the generated lyrics rhymes with the next verse, following "aabb" or "abab" rhyme sequences.

Overall, our project aims to provide songwriters and musicians with a tool that can generate high-quality and creative lyrics while adhering to the standard structure and rhyme patterns of a song. We believe this project can contribute to the fields of natural language processing and music composition, leading to exciting new possibilities for AI-generated music.

II. RELATED WORK AND BACKGROUND

In our research on lyrics generation, we refer to the work by Mateusz Modrzejewski, Jakub Szachewicz, and Przemysław Rokita, titled "Lyrics Generation Using LSTM and RNN" [1]. This study explores the use of Long Short-Term Memory (LSTM) networks and traditional Recurrent Neural Networks (RNNs) for the task of automatic lyrics generation. LSTM networks, a specialized type of RNN, are designed to remember long-term dependencies and avoid issues such as the vanishing gradient problem. They are particularly effective for sequence generation tasks like lyrics generation, as they can maintain context over longer sequences. Traditional RNNs, although simpler, tend to struggle with maintaining context over long sequences due to the vanishing gradient problem.

Another study Zihao Wang proposed an N-Gram for lyrics generation and evaluated its using perplexity and human evaluation metrics [2]. The study showed that the proposed model can generate lyrics that are grammatically correct and semantically coherent.

In addition a study in Transformers: State-of-the-Art Natural Language Processing [3] proposed a system for generating song lyrics using a transformer encoder decoder language model. The system uses the transformers for the lyric generation with a technique called "top-k sampling," which involves selecting the k most probable words at each step in the generation process and randomly choosing one of them to be the next word in the sequence to generate new lyrics. The study showed that the proposed system can generate lyrics that are comparable in quality to those written by human songwriters.

The paper "Deep Learning in Musical Lyric Generation: An LSTM-Based Approach" by Harrison Gill, Daniel Lee, and Nick Marwell, published in the Journal of Artificial Intelligence Research (JAIR) in 2021 [4], investigates the capability of LSTM networks to generate genre-specific lyrics that maintain stylistic and contextual coherence. The LSTM model is chosen for its ability to handle long-term dependencies and sequence data effectively. The training data consists of lyrics from various genres, which are preprocessed and tokenized. The model is trained to predict the next word in a sequence given the previous words, learning the patterns and structures typical of each genre. The quality of the generated lyrics is evaluated using metrics such as average line length, word variation, and thematic consistency, with human evaluators also assessing the quality, coherence, and creativity of the generated lyrics. The findings demonstrate the effectiveness of LSTM networks in producing genre-specific lyrics that are both coherent and stylistically consistent.

While these studies provide valuable insights, more advanced research is necessary to generate high-quality lyrics

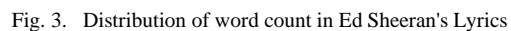
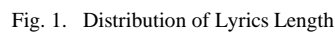
III. DATASET

The dataset used in this project is sourced from Kaggle and contains lyrics from Ed Sheeran's songs, provided in a CSV (Comma-Separated Values) format. This format ensures that the data is easily readable and accessible for analysis. The dataset consists of 179 rows and 3 columns, structured as follows:

- Each row in the dataset corresponds to a unique song by Ed Sheeran, with its title and lyrics clearly separated into distinct columns. This structured format facilitates efficient preprocessing and analysis, allowing us to train and evaluate various NLP models for lyrics generation. The comprehensive collection of lyrics provides a robust foundation for developing models that can generate new lyrics in Ed Sheeran's distinctive style.

For this project, the following text preprocessing techniques have been employed:

- These preprocessing steps will enhance the dataset's quality, making it more suitable for effective model training and generating coherent and meaningful lyrics.



The N-Gram model is one of the simplest and earliest language models used in natural language processing. It is a probabilistic model that predicts the next word in a sequence based on the occurrence of the previous N-1 words. The N in N-Gram represents the number of words considered for making the prediction. For our use case, Naïve Bayes with N-Gram language modeling with the help of Shannon's method. The final step is text generation, where we start with a seed

phrase (which can be any n-gram), and iteratively choose the next word based on the probabilities calculated. The probability is calculated based on Bayes formula given as (trigram) -tokenized text, the frequency of each n-gram token is calculated, and then for each n-gram, the conditional probability of each possible next word given the n-gram is calculated. This is done using Bayes' rule and assuming conditional independence between the n-gram and the next word.

$$P(w_n | w_{n-2}w_{n-1}) = \frac{\text{count}(w_{n-2}, w_{n-1}, w_n)}{\text{count}(w_{n-2}, w_{n-1})}$$

Fig. 4. Bayes probability for a Trigram

The N-Gram model captures the local dependencies between words, making it useful for maintaining short-term context and producing sequences that resemble the training data. However, it has limitations in capturing long-term dependencies and syntactic structures due to its fixed window size. Despite these limitations, the N-Gram model can generate coherent phrases and short lines that reflect Ed Sheeran's lyrical style by training on lyrics corpus.

B. LSTM RNN (Long Short-Term Memory Recurrent Neural Network)

RNNs, one of the many varieties of neural networks, are particularly good at jobs involving sequential data, such as voice or text. Unlike standard feedforward neural networks, which can only handle fixed-size inputs and outputs, RNNs can process variable-length data sequences. They can mimic dependencies between different sequence components because they operate by maintaining an internal state that is altered by each input in the sequence. Because lyrics usually have a repeating structure, such verses and choruses, RNNs can be used to simulate the lyrics' structure in the lyric generation instance.

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to address the limitations of traditional RNNs, such as the vanishing gradient problem. Introduced by Hochreiter and Schmidhuber in 1997, LSTMs can maintain long-term dependencies by using a gating mechanism to control the flow of information.

The LSTM architecture consists of a series of gates: the input gate, forget gate, and output gate, which regulate the addition and removal of information to the cell state. This structure allows LSTMs to retain relevant information over long sequences and discard unnecessary information. The key components of an LSTM cell are as follows:

- Forget Gate: Decides what information to discard from the cell state.
- Input Gate: Decides which new information to add to the cell state.
- Cell State Update: Updates the cell state with new information
- Output Gate: Decides what information to output.

LSTMs can be used to model the structure of the lyrics in the context of lyric generation while also enabling the network to retain crucial information from earlier lines or verses.

C. GPT2 Transformer

GPT-2 is a Transformer-based model trained for language modelling. Unlike RNNs and LSTMs, transformers process data in parallel and capture relationships between all elements in a sequence simultaneously. Transformers do not rely on recurrence but instead operate on self-attention. Self-attention allows the model to weigh the importance of different input tokens when making predictions, enabling it to capture long-range dependencies without the need for sequential processing. The parallel processing capability of transformers also significantly reduces training time and computational resources compared to sequential models.

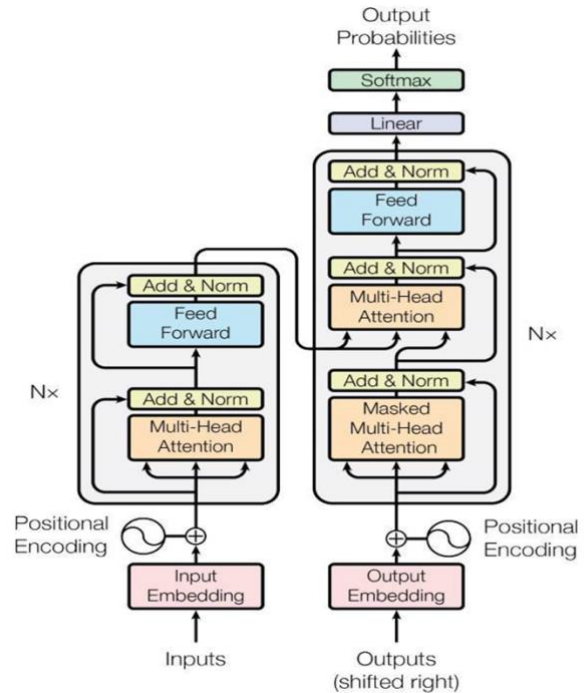


Fig. 5. Transformer architecture

The Transformer architecture composed of an Encoder-Decoder structure with the following key components:

Encoder:

- Multi-Head Self-Attention: Captures dependencies between words by attending to all positions in the input sequence simultaneously.
- Position Embeddings: Adds positional information to the input embeddings to account for the order of words, as the self-attention mechanism itself does not consider order.
- Feedforward Network: Applies a position-wise fully connected feedforward network to each position separately and identically.
- Non-Linearity and Normalization: Uses layer normalization and activation functions (like ReLU) to stabilize and introduce non-linearity to the model.

Decoder:

- Multi-Head Cross-Attention: Attends to the encoder's output while generating each word in the

output sequence, allowing the decoder to focus on relevant parts of the input.

- **Masked Self-Attention:** Ensures that the prediction for a position only depends on earlier positions, which is crucial for autoregressive generation.

GPT-2 will generate lyrics by predicting the next word in a sequence based on the context provided by the previous words. GPT-2 leverages its attention mechanism to focus on relevant parts of the input text, enabling it to capture both local and global context. During the fine-tuning process, the model learns the patterns, structure, and style of the song lyrics in the training dataset. When generating lyrics, GPT-2 takes an initial prompt or seed text (a few words or lines) as input and then iteratively predicts the most likely next word, appending it to the input sequence. This process continues, with each new word being influenced by the preceding words, until the model generates a complete set of lyrics or reaches a specified length. The generated lyrics reflect the patterns and themes it learned during fine-tuning, allowing GPT-2 to create coherent and stylistically appropriate song lyrics.

V. EVALUATION METHODS

A. Perplexity

Perplexity is a widely used evaluation metric in text generation tasks, particularly for language models like GPT. It measures how well a model predicts a sample of text by assessing the probability it assigns to the true sequence of words. Perplexity is defined as the exponentiation of the average negative log-likelihood of a sequence, meaning it reflects the model's uncertainty in its predictions. A lower perplexity score indicates that the model is more confident and accurate in predicting the next word in a sequence, implying better performance. In contrast, higher perplexity suggests that the model is less certain and potentially generating less coherent or relevant text.

B. ROUGE Score

ROUGE score is a set of metrics commonly used to evaluate the quality of text generation, especially in tasks like summarization and translation. ROUGE compares the overlap of n-grams, word sequences, and word pairs between the generated text and one or more reference texts. The most frequently used ROUGE metrics are ROUGE-N (which measures n-gram overlap), ROUGE-L (which considers the longest common subsequence), and ROUGE-S (which looks at skip-bigram overlap). A higher ROUGE score indicates that the generated text more closely matches the reference text, implying better performance. ROUGE is particularly valued for its ability to capture recall, making it effective in evaluating whether a model generates comprehensive and relevant content.

C. BLEU Score

BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of text generation, particularly in machine translation. It measures how closely the generated text matches one or more reference texts by calculating the precision of n-grams (sequences of words) in the generated

output. BLEU accounts for both the number of matching n-grams and the brevity of the generated text, with a penalty applied to overly short translations to ensure meaningfulness. A higher BLEU score indicates better alignment with the reference text, suggesting more accurate and fluent generation. However, BLEU has limitations, such as its sensitivity to exact word matches and its potential to overlook semantic meaning, making it important to complement it with other evaluation metrics for a more comprehensive assessment of text quality.

D. Cosine Similarity

Cosine similarity measures the cosine of the angle between these vectors, resulting in a value between -1 and 1. A value of 1 indicates that the texts are identical in terms of direction, meaning they are highly similar, while a value of 0 suggests no similarity, and -1 indicates complete opposition. Unlike other metrics that focus on exact word matches, cosine similarity captures the overall context and meaning, making it valuable for assessing the semantic closeness of generated text to a reference.

VI. RESULTS / CONCLUSION

A. N-GRAM Model

We define the N for N-gram. In our case, we have used N = 3 (trigram). Once we have the N-gram tokens, we calculate the probability of each token appearing, using Bayes rule. From the dataset, we observe that each sentence in the song is approximately 5-8 words long. After every 5-8 words, a new phrase or sentence typically appears in the song. Usually, the last word of each sentence rhymes with the last word of the next sentence. To distinguish these ending words, we append "/n" to the word. For example, the words "lover/n" and "lover" will be treated as different words by the tokenizer. This method is used to separately capture ending words, as they tend to rhyme with the last word of the next sentence.

Training and Validation Perplexities:

Average Train Perplexity: 2701.0551

Average Validation Perplexity: 2711.4854

Fig. 6. Perplexity of N-Gram Model

ROUGE-1: Precision: 0.1250, Recall: 0.1000, F1: 0.1111
 ROUGE-2: Precision: 0.0000, Recall: 0.0000, F1: 0.0000
 ROUGE-L: Precision: 0.1250, Recall: 0.1000, F1: 0.1111

Fig. 7. Rouge Score of N-Gram Model

BLEU score: 0.0120

Fig. 8. BLEU Score of N-Gram Model

The high perplexity values for both training (2701) and validation (2711) suggest that the model struggles to predict the next word in a sequence, indicating poor generalization and potential overfitting. The BLEU score of 0.0120 is extremely low, showing that the generated text is far from the reference text. Additionally, the ROUGE scores (ROUGE-1 precision of 0.1250, ROUGE-2 precision of 0.0, and ROUGE-L precision of 0.1250) further emphasize that the model's output has limited overlap with the reference text in terms of n-gram and longest common subsequence matches. Overall, the model performs poorly in generating coherent and contextually relevant lyrics, indicating that it requires significant improvement or the use of more advanced models like RNNs, Transformers, or pre-trained language models for better performance.

B. LSTM RNN Model

The model begins with an Embedding layer that converts word indices into dense vectors of fixed size. This is followed by a Bidirectional LSTM layer, which processes the sequences in both forward and backward directions to capture dependencies from both ends. Finally, a Dense layer is used to produce the output, representing the probability distribution over the vocabulary for the next word in the sequence. The output of the Dense layer is passed through a Softmax activation function to produce probabilities. The model is optimized using the Adam optimizer.

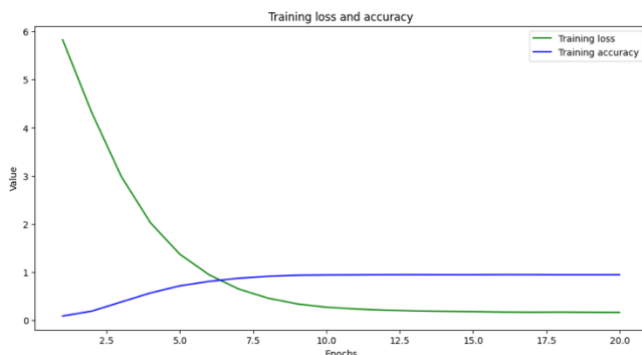


Fig. 9. Training Loss and Training Accuracy of RNN LSTM Model

The model improves both in training and validation metrics, indicating good learning in early epochs. In later epochs, training loss continues to decrease, and training accuracy continues to increase, while validation metrics (loss and accuracy) do not show corresponding improvements. Moreover, we observed validation accuracy plateaus and does not show significant improvement, indicating that the model's performance on unseen data does not improve as it continues to train and appears to be overfitting the training data.

ROUGE Scores:
ROUGE-1: 0.6252
ROUGE-2: 0.1957
ROUGE-L: 0.6167

Fig. 10. ROUGE Score of RNN LSTM Model

Lyrics Generated: two mind loves one kind drifting edge fall fall fall would fall learning speak kisses cheeks lif
ted edge fall fall fall fall would fall fall fall fall would fall
Perplexity: 26867374.73446718

Lyrics Generated: start something beautiful start something new one make lose start something new ill throw away w
atch fall arms ill throw away watch fall earth stand upon words sing ive thrown away watched fall arms ive thrown
away watched fall take back take home watch fall earth take back start something beautiful start something new
Perplexity: 3490.914193468222

Lyrics Generated: tell youd turn man asks hand cause youre waiting know youre gonna away ive got plans leave woul
d take away hopes dreams stay senses come life im stumbling home drunk ever ill never leave cause one friends gone
find another place let hearts collide promise youll never leave cause one take hand heart soul eyes know everythin
g changes well strangers see could stay within walls bleed stay senses come life im stumbling home drunk ever ill
never leave cause one friends gone find another place let hearts collide promise youll always friend cause one stu
mbling half drunk getting lost gone tell way home listen sad songs singing love goes wrong senses come life im stu
mbling home drunk ever ill never leave cause one friends gone find another place let hearts collide promise youll
always friend cause one
Perplexity: 7132.67697425221

Fig. 11. Perplexity of RNN LSTM Model

The LSTM RNN model used for lyrics generation exhibits mixed performance based on the metrics. The perplexity scores vary significantly, with one being exceptionally high at 26,867,374.73, and others more reasonable at 3,490.91 and 7,132.68. The high perplexity suggests that the model may have difficulty predicting the next word in certain contexts, leading to inconsistent generation quality. However, the ROUGE scores—ROUGE-1 at 0.6252, ROUGE-2 at 0.1957, and ROUGE-L at 0.6167—indicate that the model is relatively effective in capturing n-gram overlap and maintaining the structure of the reference text. This suggests that while the model may struggle with some aspects of language modeling (as evidenced by the perplexity), it still manages to generate text with a reasonable degree of similarity to the reference lyrics in terms of content and structure. Further fine-tuning or adjustments may help improve its performance, particularly in reducing perplexity across all contexts.

C. GPT2 Transformer Model



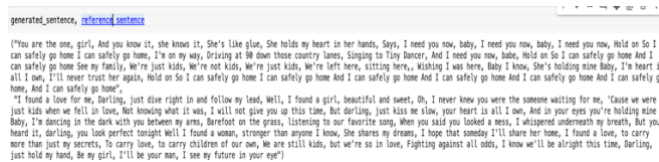
Fig. 12. Training and Validation Loss of GPT2 Transformer Model

Train Perplexity: 1.1086199390533467
Validation Perplexity: 2.426796710223497

Fig. 13. Training and Validation Perplexity of GPT2 Transformer Model

The training loss steadily decreases from 1.2164 at epoch 1 to 0.2026 at epoch 5, highlighting the model's enhanced ability to predict sequences accurately during training. Similarly, validation loss improves from 1.2973 to 0.8866, indicating that the model generalizes well to new, unseen data. Perplexity, a key measure of model performance, reveals that the train perplexity is 1.1086, and the validation perplexity is 2.4268. While the train perplexity is quite low, suggesting excellent fit on the training data, the higher validation perplexity indicates room for improvement in generalization.

We also used cosine similarity and ROUGE Score on reference text.



The screenshot shows a text comparison interface. At the top, there are two tabs: 'generated sentence' and 'reference sentence'. The 'reference sentence' tab is selected. Below the tabs, the reference text is displayed: "You are the one, girl, And you know it, she knows it, She's like glue, she holds my heart in her hands, Says, I need you now, baby, I need you now, baby, I need you now, Held on So I can safely go home I can safely go home, I'm on my way, Driving at 98 down those country lanes, Singing to Tiny Dancer, And I need you now, babe, Held on So I can safely go home And I can safely go home See my family, We're just kids, We're not kids, We're just kids, We're left here, sitting here, Wishing I was here, Baby I know, She's holding mine Baby, I'm heart I all I see, I'll never trust her again, Held on So I can safely go home I can safely go home And I can safely go home And I can safely go home And I can safely go home And I can safely go home, And I can safely go home". Below the reference text, the generated text is displayed: "I found a love for me, Darling, just dive right in and follow my lead, Well, I found a girl, beautiful and sweet, Oh, I never knew you were the someone waiting for me, 'Cause we were just kids when we fell in love, Not knowing what it was, I will not give you up this time, But darling, just kiss me slow, your heart is all I own, And in your eyes you're holding mine Baby, I'm dancing in the dark with you between my arms, Barefoot on the grass, Listening to our favorite song, When you said you looked a mess, I whispered underneath my breath, But you heard it, darling, you look perfect tonight Well I found a woman, stronger than anyone I know, She shares my dreams, I hope that someday I'll share her home, I found a love, to carry more than just my secrets, To carry love, to carry children of our own, We are still kids, but we're so in love, Fighting against all odds, I know we'll be alright this time, Darling, just hold my hand, Be my girl, I'll be your man, I see my future in your eyes".

Fig. 14. Reference text and Generated text from model

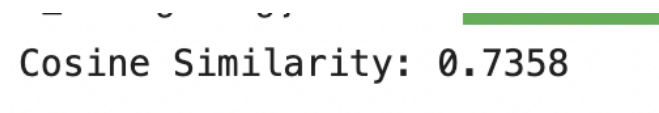
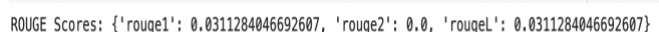


Fig. 15. Cosine Similarity of GPT2 Transformer Model on reference text



The screenshot shows the ROUGE Scores output: "ROUGE Scores: {'rouge1': 0.0311284046692607, 'rouge2': 0.0, 'rougeL': 0.0311284046692607}".

Fig. 16. ROUGE Score on reference text and generated text from model

The ROUGE scores indicate very low recall for both ROUGE-1 and ROUGE-L, with values of 0.0311. This suggests that the generated text has minimal overlap with reference texts in terms of n-grams and longest common subsequences, implying that while the text may be contextually coherent, it may lack substantial content similarity or alignment with the reference data. ROUGE-2, which measures the overlap of bigrams, shows a score of 0.0, reflecting that there is no significant overlap at this level.

On the other hand, the cosine similarity score of 0.7358 suggests a reasonable degree of similarity between the generated and reference text in terms of their vector representations. This indicates that while the specific content

might not align closely with the reference, the overall semantic content and structure are somewhat similar.

But since both are different lyrics and we got cosine similarity good means they are similar to each other but since rouge score is less suggesting they are different song lyrics.

In conclusion, comparing all models we saw N-Gram is not the best choice for lyrics generation and RNN performed better than N-Gram and was relatively effective in capturing n-gram overlap and maintaining the structure of the reference text. This suggests that while the model may struggle with some aspects of language modeling (as evidenced by the perplexity), it still manages to generate text with a reasonable degree of similarity to the reference lyrics in terms of content and structure. GPT2 Transformer was best among all the three models with best perplexity and cosine similarity. Although, it has some overfitting issue suggesting room for improvement and we can use more data to train and utilize ablation methods like regularization to reduce overfitting of the model.

REFERENCES

- [1] Mateusz Modrzejewski, Jakub Szachewicz, and Przemysław Rokita, "Lyrics Generation Using LSTM and RNN."
- [2] Zihao Wang. *N-gram and LSIM based Language Models*. Research School of Computer Science, the Australian National University.
- [3] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara May Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, et al. 2020. *Transformers: State-of-the-Art Natural Language Processing*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38- 45, Online. Association for Computational Linguistics.
- [4] Harrison Gill, Daniel Lee, and Nick Marwell, "Deep Learning in Musical Lyric Generation: An LSTM-Based Approach".