The Lyrical Empiricals: *Deep Learning for Lyric Generation & Unsupervised Topic Modeling*

By: Josh Horner, Yanying Jiang, & Kenny Tang

Introduction & Project Statement:

Music and songwriting are forms of expression to an artist. Similarly to poetry, musical lyrics encapsulate the thoughts and emotions of a writer; and those thoughts are reflective of the state of the world and the time in which a song was composed. Historical events that are significant enough to define an era necessarily influence and provide inspiration to musical artists. During the era of the Vietnam war, many songs bore themes of peace, heroism, and/or patriotism. "Give Peace a Chance," written by John Lennon in 1969, became a timely anthem calling for peace and preaching pacifism during the war in Vietnam. Similarly, "Ohio," written by Neil Young in 1970, recounted the slaying by National Guardsmen of four students at an anti-war protest. Overall, there is a common theme between the songs written within a similar era. Common themes are not only found in historical events, but also modern times. Within the past decade, there has been a surge in musical themes that discuss Black Lives Matter and the empowerment of women. In theory, the close relationship between a song and its historical period of origin should be detectable by an analysis of the lyrics. The main challenges to such an effort include the retrieval of accurate labels and a variety of confounding variables, such as genre and authorship. With high quality data and the right methods, we believe that lyrical data from the past century's popular music can offer valuable insight into our history and culture.

Our project has two major components. The first component follows the ideas of our previous statements; that is, we attempt to cluster songs in some form using Natural Language Processing, referred to as NLP throughout the rest of the report. For the second component, we decided to add a playful flavor to our project by building generative models that produce lyrics using the unique writing signature of specific music artists. For the 32 American artists in our dataset who have recorded the most songs, we built generative models for each using neural networks. To complement our generative models, we created topic models to observe keywords and topics that give insight into an artist's dominant writing style.

Data:

Our dataset was obtained through Kaggle. In our dataset, we were given 379,919 entries, each representing a single song. For each song, we were given the associated artist name, album name, lyrics, and language in which the lyrics were written. Since our team would be unable to make sense of songs written in any foreign language, we decided to keep only songs written in English. After removing all songs with non-English lyrics, our dataset contained 191,804 songs.

For the purpose of the Generative modeling component of our project, we decided to extract a much smaller subset of this data. We isolated songs belonging to the top 32 artists, those with the most recorded songs and are American. The list of these artists can be found below. This subset of our data had 14,328 songs across 32 artists. We used this more focused dataset to conduct further exploratory analysis and sanity checks to identify duplicates and other potential issues. One particular issue we found with our dataset was the heavy noise associated with songs that feature other artists. Because our goal is to generate lyrics using a specific artist's writing/speech patterns, songs that contain lyrics sung by another artist would introduce patterns uncharacteristic of the target artist. This would degrade our models, causing them to generate inaccurate and unrepresentative lyrics. We used Regex to find and remove songs where the title indicated other artists were featured. Furthermore, our exploration discovered songs that had identical lyrics for multiple entries. Upon further inspection, we found that some of these entries were raw duplicates (i.e. identical songs with slightly different song titles), some of these entries were famous songs written by other artists and covered by multiple artists in our dataset (i.e. famous christmas carols covered by multiple artists), and the rest of the the duplicate entries were caused by songs that were collaborated by multiple artists in our dataset (i.e. In a collaboration between Chris Brown and Eminem, each received credit via multiple entries in our dataset). We excluded all of these songs to collectively deal with these issues. The resulting dataset for our generative model has 13,947 songs across 32 artists. From these 32, we selected Eminem and Taylor Swift for the examples in this paper.

In addition to our primary song dataset, we were also provided with a dataset that contained meta data for the artists. For each artist, we were given features describing the three most fitting genres, the artist's song count, and an artist popularity score. The only feature we extracted from this dataset was the genres, which we merged with our song dataset.

Finally, we cleaned the text of the lyrics themselves. Lyric cleaning proved to be one of our biggest challenges as the text was extremely messy. Because the lyrics were attained via crowdsourcing, they were recorded in wildly inconsistent formats. We again used Regex to remove annotations labeling the sections of a song. These annotations were easy to capture as they tended to be enclosed in brackets, parentheses, or curly brackets (i.e. [chorus] or {verse1}). We also identified and removed as many ad libs as we could. These were primarily encased between stars or double stars (i.e. *clap* or **whistle**). In addition, we discovered incorrect spellings of certain words that subtly changed their meaning from colloquial nicknames to racial slurs. Through some research we confirmed what the proper word should have been and replaced them accordingly. Lastly, we censored various expletives by creating a list of profane words and using Regex to replace all but the first character with stars (i.e. f***). The lyrics suffer from other issues, including some misspelled words; however, we decided to leave them given their relative infrequency, and our inability to observe and replace all misspelled words.

For the purpose of our clustering models, an important feature that was not provided to us by our Kaggle dataset was the year in which each song was written. Because we had initially intended to cluster songs by musical eras, we would have liked to have had song years to act as ground truth labels for evaluation purposes. Using various API methods, we were able to extract years for a subset of our songs through MusicBrainz and Spotify. Unfortunately, these extracted years were inaccurate, yielding such ridiculous examples as songs written in the late 2010s attributed to Elvis Presley (and many other long-deceased artists). Some of these likely reflected the dates in which a remastered version was released. We have made attempts to correct these errors by identifying outliers and using regression models to predict the correct years of these outliers. In the end, however, we were unable to acquire accurate results that would allow us to cluster and evaluate our decades with confidence. After weeks of our team's diligent efforts to extract years, we decided to forgo the idea and proceed further in our project with the dataset's native features.

Top 32 Artists selected for our Generative Model:
Frank Sinatra, Elvis Presley, Dolly Parton, Lil Wayne, Chris Brown, Guided by Voices, Prince, Johnny Cash, Bob Dylan, George Jones, Neil Young, Bruce Springsteen, Snoop Dogg, Eminem, 50 Cent, Roy Orbison, Ella Fitzgerald, Taylor Swift, Waylon Jennings, 2Pac (Tupac Shakur), BB King, Bon Jovi, George Strait, Madonna, Diana Ross, Bill Monroe, Beach Boys, Barry Manilow, Alice Cooper, Nas, Ray Charles, Beck

Genre Clustering:

One remaining avenue to explore was the question of how genres are characterized by their songs' lyrics. Genres tend to be thought of in terms of their "sound" and instrumentation. However, it is reasonable to assume that, as a genre develops and differentiates musically, it will also develop its own lyrical themes and patterns. These patterns might be discernible through an analysis of relative word frequency. While the primary song dataset contains no genre labels, the artist dataset includes a field with the 3 genres deemed most descriptive of the artist's collective work. Using these *artist* genres as proxies for the individual *song* genres seems imperfect, but reasonable.

The first problem was how to represent the genres. The data contained 28 unique "genre sets," which would prove a bit unwieldy for focused analysis and visualization. Since many of these genre sets shared individual genres in common, we decided to represent them as a relatively smaller number of clusters. We accomplished this by tokenizing and vectorizing the genre sets, effectively transforming them into a one-hot encoding scheme. The frequency vectors were then clustered, based on an elbow score, into 8 classes against which the lyric clusters could be compared.

The exercise ultimately proved uninformative, as there appeared to be little correspondence between the genre clusters and clusters derived from the lyrics themselves. Despite attempting several vectorization approaches and various values of k for the lyric tokens, t-SNE projections were unable to identify significant separation between classes and also failed to reveal any obvious correlation between the genre and lyric clusters. The t-SNE graph below (Figure 1) shows the thorough diffusion of clusters, a fact that is reinforced by the heterogeneity of the stacked bar chart (Figure 2).

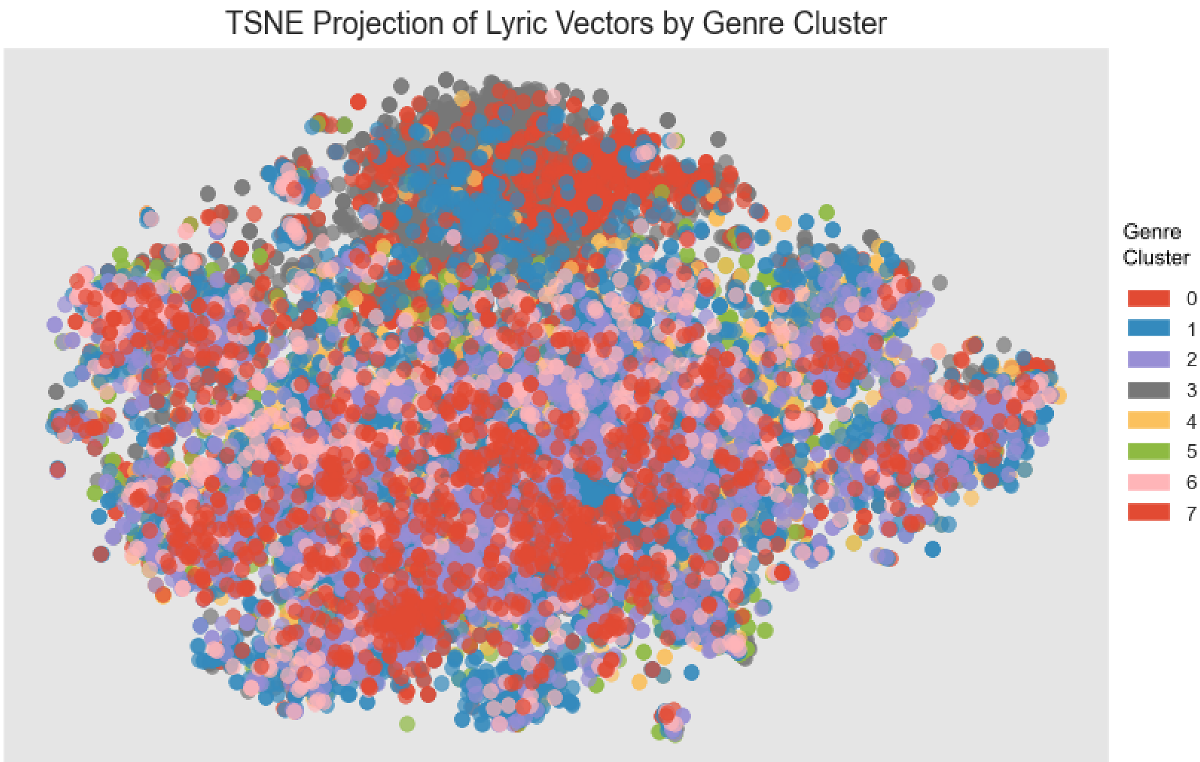

Figure 1

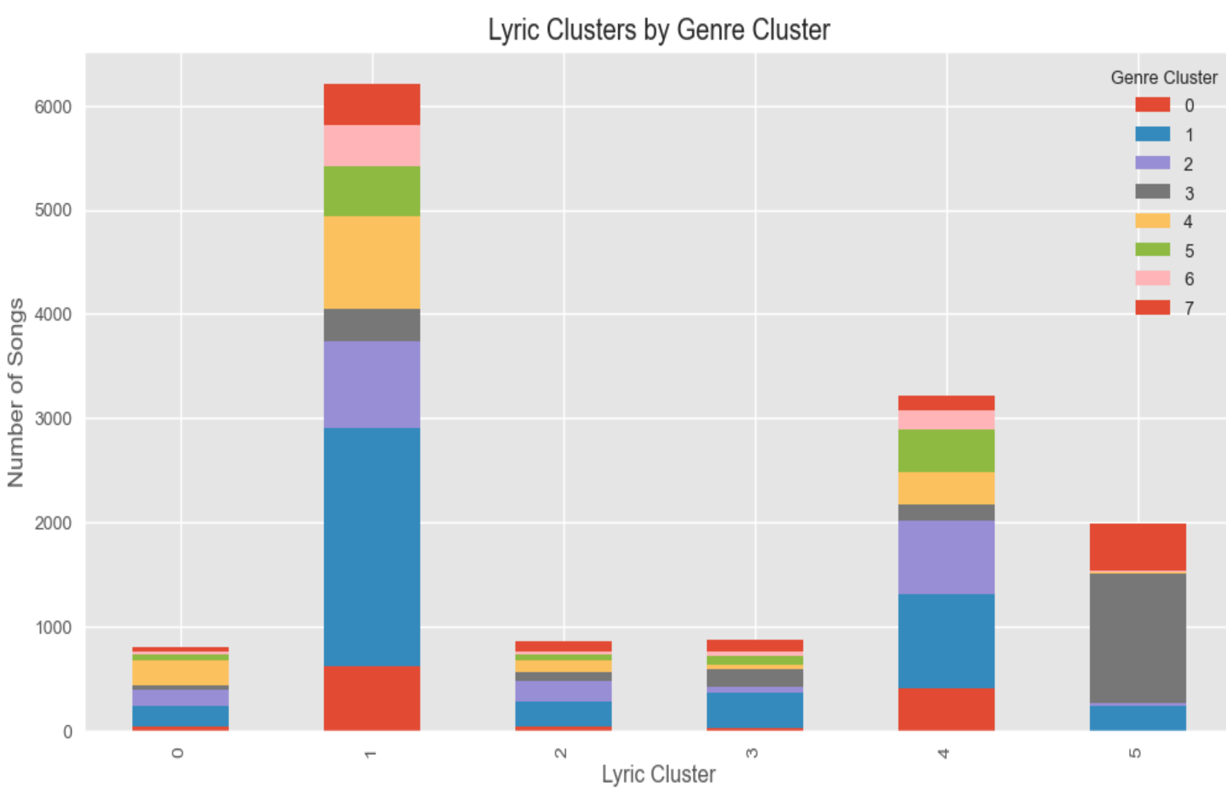

Figure 2

Generative Model Methods:

Our next goal is to build a generative model for song lyrics that can learn the patterns and structures that exist within a corpus of existing song lyrics, and then use this knowledge to generate new lyrics that are similar in style and content to the original corpus.

Special data preprocessing: There were a few preprocessing steps completed for this task. We aimed to compile a list of the top 50 artists based on their song count. However, certain artists were excluded due to either not being from the United States or being a music group with multiple artists. As a result, our refined list features 32 exceptionally prolific US artists (please refer to the list of artists in the data section). We filtered the dataset in this manner to obtain a more cohesive subset for training purposes, thereby enabling the model to better learn the underlying structure and patterns.

It's worth mentioning that we did not proceed with stopword removal or lemmatization, since in some cases, stopwords may carry important contextual information and contribute to the overall meaning and tone of the lyrics. Some artists may use certain stopwords more than others, and we hoped to see that pattern in the generated lyrics. Lemmatization can also result in the loss of some important information. For example, "loving" and "loved" have different meanings and may be used in different contexts, so reducing both of them to "love" may lead to some loss of sentiment.

Choice of model: Inspired by the TensorFlow tutorial on text generation(TensorFlow, 2023), we decided to go with a character-based text generation model using RNN(Recurrent Neural Network). RNN is a commonly used language model due to its capability of predicting sequences of words. These types of models are also referred to as sequence to sequence models.

Traditionally, RNNs take in a sequence of words as input, and then process one word at a time while considering the context of preceding words in the sequence. In doing so, the model can learn the dependencies between neighboring words. The referenced tutorial takes a slightly different approach in that it is character-based. It utilizes a dataset of Shakespeare's writing, and uses a given sequence of characters from the data to predict the next character in the sequence, instead of predicting the next word. This methodology uses tokens to represent individual characters rather than words, and the corpus consists of all unique characters found within the dataset. In the models we used, the input text contains all characters in a sequence except the last character, and the target text contains all characters in the sequence except the first character.

More specifically, the two models we chose are GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory). They are both proficient at handling sequential data and capturing long-range dependencies, and they both address the vanishing gradient problem in traditional RNNs. We wanted to compare the two models and see which one performs better under our evaluation metrics.

GRU: The model has three main components: an embedding layer, a GRU layer, and a dense layer, with vocab_size, embedding_dim, and rnn_units as its initial parameters.
The embedding layer, also called the “input layer,” maps the input character indices to dense vectors of fixed size (embedding_dim). In this way, it helps the model learn meaningful representations for each character.
The GRU layer uses rnn_units as the number of recurrent units, and is responsible for learning the sequential patterns in the input data. This layer maintains a hidden state that is updated at each time step, and this hidden state serves as a memory that encodes the information from the previous character in the sequence.
The dense layer uses vocab_size as its output units, and it is used to produce predictions for each character in the vocabulary.

LSTM: The model has three main components as well: an embedding layer, an LSTM layer, and a dense layer. Therefore, we replaced the GRU layer with the LSTM layer here.

With the input and output layer being the same as GRU model, the LSTM layer also used rnn_units as the number of recurrent units and is also responsible for learning the structure of input sequence, then it returned both the output sequences and the final hidden state as well as the cell state.

Training the model: We trained each instance with the complete lyrical corpus of the corresponding artist as input text files, splitting the input data into training and validation sets with a ratio of 0.8:0.2, so as to ensure the model performs well on unseen validation datasets and to prevent overfitting. For each instance, we generated 10 sets of lyrics. We also utilized Tensorboard to visualize the trend of loss and perplexity during the training progress.

Parameter tuning: Parameter tuning was conducted through random search, focusing on learning rate, embedding_dim, and rnn_units, with the goal of minimizing the perplexity. During the training, we observed that increasing the epoch size substantially improves model performance up to 40 epochs; however, the rate of improvement becomes flatter beyond 40 epochs.

The params tuned are:

- Embedding dimension: The size of the word embeddings. We searched the range of [64,128,256].
- RNN units: The number of recurrent units in the GRU layers. We tuned the range of [128,256,512,1024].
- Learning rate: The step size taken by the optimizer during training. We searched a range of [1e-4, 1e-2]

There are a few params that we used the default value:

- Optimizer: The optimization algorithm used for training - We used Adam

- Batch size: The number of samples used in each update during training - We used 64
- Number of GRU layers: the number of stacked GRU layers. Stacking more layers can increase the risk of overfitting and cost more resources, thus we decided to go with a single layer.

Param tuning for GRU:
According to the above method, we obtained the below best parameters for use in final training.

Best perplexity: 35614.50
Best param - embedding_dim: 256
Best param - rnn_units: 1024
Best param - learning_rate: 0.0008328

Param tuning for LSTM:
Since the character-based LSTM model is used as a comparison of GRU, we used the same input and the same strategy of training and the same method of parameter tuning has been applied.

Best perplexity: 654548.0
Best param - embedding_dim: 128
Best param - rnn_units: 1024
Best param - learning_rate: 0.00040995

Generating the lyrics:
We utilized ten distinct self-composed seed-text lines as the starting point for both models. These lines were used to generate ten unique songs for each artist, allowing us to observe how the generative model adapts to the respective artist's style based on a common foundation. The seed-text lines are as follows:

1. "In the land of the free,"
2. "On a starry night,"
3. "Through the city streets,"
4. "Chasing dreams and memories,"
5. "As the sun goes down,"
6. "Lost in the rhythm of life,"
7. "With a heart full of hope,"
8. "In the shadows of skyscrapers,"
9. "Under the neon lights,"
10. "Where the music never stops."

By using the same seed-text lines for both models, we aim to evaluate the capacity of our generative model to create lyrics that are consistent with each artist's distinctive style.

Evaluation for Generative Model:

Cross-entropy loss and perplexity:

In both GRU and LSTM models, the loss function we chose is cross-entropy loss. We also customized a function to calculate the perplexity of the model. Perplexity is a commonly used metric in language modeling tasks that measures how well the model makes predictions. It is defined as the exponentiation of the cross-entropy between the true labels and the model's predictions. A lower perplexity indicates a better model. During our training, we monitored our progress with the goal of minimizing the cross-entropy loss and the perplexity.

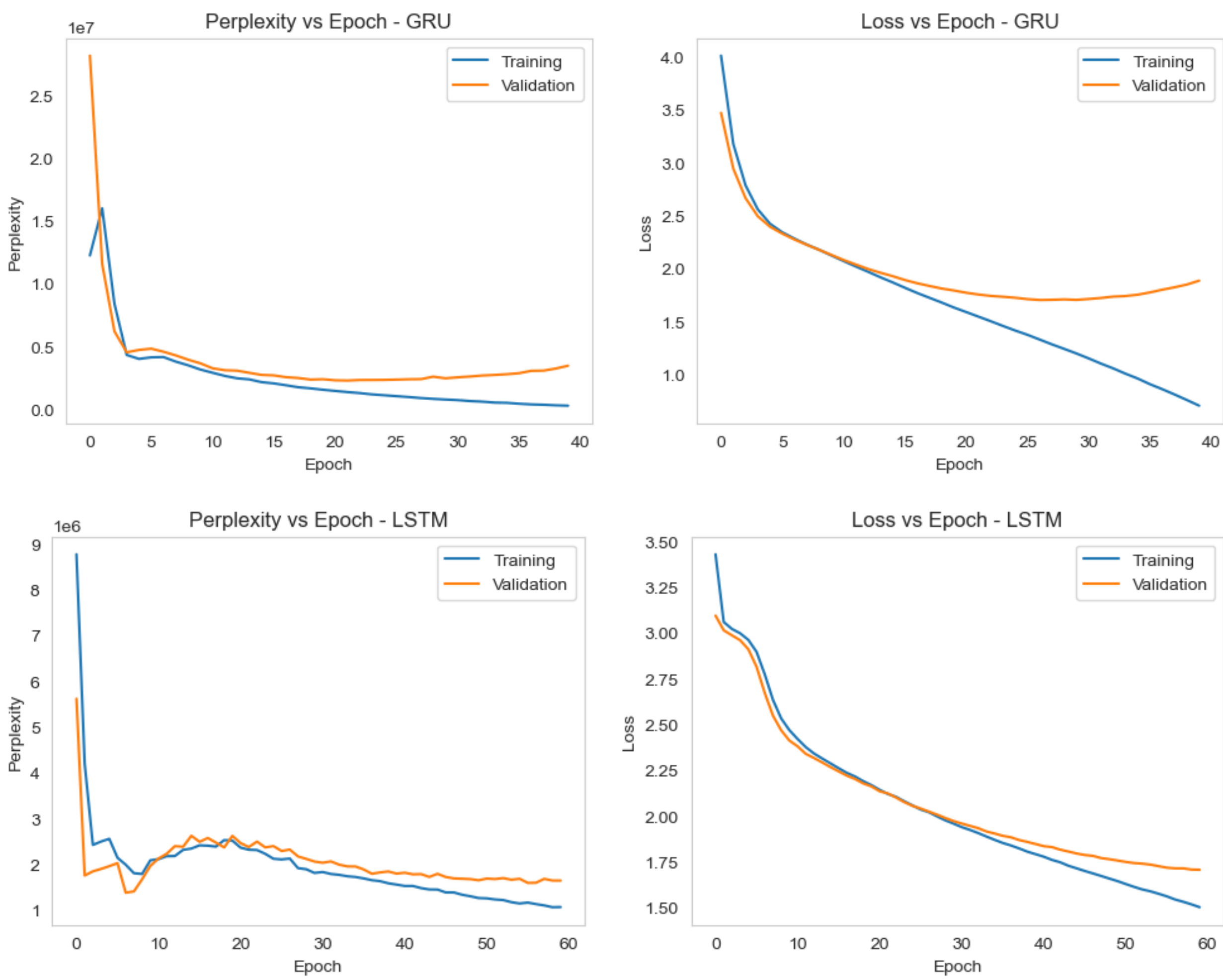

Figure 3

During the training of GRU and LSTM models, we observed that increasing the number of epochs does not always result in improved performance. On average, after around 35 epochs, increasing the number of epochs may lead to a decline in model performance on validation datasets, as demonstrated in the plots above. This trend was also evident when training the LSTM model with 60 epochs.

This phenomenon can be attributed to overfitting, which occurs when our model becomes excessively familiar with the training data and captures too much noise and random variations from the training dataset. As a result, the model's performance on unseen dataset declines.

To mitigate overfitting, we utilized the above plots generated in TensorBoard to monitor the model's performance on both training and validations datasets. We adopted early stopping as the technique for mitigation, in that we halted the training process when the performance on the validation set started to decrease in order to avoid the negative impact of excessive training. We ended up using an epoch size of 40 for LSTM as well, instead of 60.

Custom evaluation metrics define:
Evaluation has always been a difficult aspect for text generation tasks. Metrics like perplexity, which measures "randomness" or BLEU(BiLingual Evaluation Understudy) which measures similarity based on the overlap of n-grams, do not accurately measure coherence, creativity, or the natural structure of the generated lyrics. Thus we decided to introduce additional custom metrics for evaluation.

We adapted some ideas from the study "Deep Learning in Musical Lyric Generation: An LSTM-Based Approach" (Gill, Lee, & Marwell, 2020), which describes a few custom evaluation metrics for generated text. Based on their research, we implemented and modified the following evaluation methods to suit our specific needs.

1. *Vocabulary Overlap*: This metric measures the average Jaccard similarity between the vocabulary used in a song and the vocabularies of other songs by the same artist. Lower values indicate less overlap and greater diversity in the artist's vocabulary.
2. *Word Variation*: This metric measures the diversity of the vocabulary used in a song by calculating the ratio of unique words to the total number of words in the song. Higher values indicate a more diverse vocabulary and more creativity.
3. *Song Length*: This metric measures the total number of words in a song, providing a simple way to compare the lengths of generated versus real songs.
4. *Average Line Length*: This metric calculates the average number of words per line in a song, which can give an indication of the song's structure.
5. *Count Word Repetition*s: This metric counts the number of consecutive word repetitions in a song, indicating the presence of repetitive phrases or patterns in the lyrics, which is a strong representation of artists' unique styles.
6. *Point of View*: This metric calculates the difference between the number of lines starting with "I" and the number of lines starting with "you" as a percentage of the total number of lines in the song. This provides an indication of the predominant point of view in the lyrics.

Below is an example of how these metrics work on the artist Eminem:

Summary of Evaluation Metrics on Artist: Eminem

Song Type	Vocabulary Overlap	Word Variation	Song Length	Average Line Length	Word Repetitions	Point of View
GRU	0.0139	0.6470	211.6000	8.0860	0.2000	10.8260
LSTM	0.0140	0.6970	205.6000	8.2730	0.5000	6.8690
Real	0.0142	0.4711	653.5293	8.2297	4.2875	7.5522

1. Vocabulary Overlap: Generated lyrics have similar vocabulary overlap with the real ones.
2. Word Variation:Generated lyrics show slightly higher word variation than real lyrics, indicating more diverse vocabulary use, which may result from the models generating non-existing or uncommon words. LSTM generated lyrics have even higher variation than GRU generated lyrics, which aligns with our observations that LSTM generates more random non-existing words.
3. Song Length: Both GRU and LSTM generated lyrics are significantly shorter than real lyrics. Note that this comes from the way we built the model: generated text is set to a fixed length of 1000 characters. We wanted to keep this metric to see the characteristics of the original work.
4. Average Line Length: Generated lyrics have similar line length with the real lyrics, indicating that our model performs well on learning this aspect.
5. Word Repetitions: Generated lyrics have fewer word repetitions than real lyrics, indicating less repetition. LSTM generated lyrics have slightly more repetitions than GRU generated lyrics.
6. Point of View: GRU generated lyrics have a higher point of view score, suggesting more first-person perspective, while LSTM generated lyrics have a closer point of view score to the real lyrics.

And we summarized the same for Taylor Swift:

Summary of Evaluation Metrics on Artist: Taylor Swift

Song Type	Vocabulary Overlap	Word Variation	Song Length	Average Line Length	Word Repetitions	Point of View
GRU	0.0160	0.5120	224.4000	6.6490	21.7000	4.9200
LSTM	0.0160	0.6670	212.9000	6.6540	1.0000	7.0830
Real	0.0160	0.3738	319.1750	6.6476	4.6194	6.2835

1. Vocabulary Overlap: Both generated and real lyrics have the same overlap.
2. Word Variation: Generated lyrics show higher word variation, indicating a more diverse vocabulary. LSTM generated lyrics have even higher variation than GRU generated ones.

3. Song Length: Generated lyrics are shorter than real lyrics due to the model setup.
4. Average Line Length: Generated and real lyrics have nearly identical average line lengths.
5. Word Repetitions: GRU generated lyrics contain more word repetitions, indicating much higher repetition. In contrast, LSTM generated lyrics have fewer repetitions than real lyrics.
6. Point of View: GRU generated lyrics have a lower point of view score, suggesting less first-person perspective, while LSTM generated lyrics demonstrate a greater first-person perspective compared to the original songs.

The above metrics provide a comprehensive overview of various aspects of song lyrics, such as vocabulary diversity, structure, and point of view. By summarizing these metrics, we are able to gain insights into the overall characteristics of an artist's lyrics and compare our generated lyrics to the original work of the same artist. We learned that word variation, word repetition and point of view are the top 3 metrics that show the largest differences between generated songs and original songs, and that our model does well for vocabulary overlap and average line length in terms of imitating the original style. To further explore the results, we have created some additional visualizations for two sample artists, Eminem and Taylor Swift.

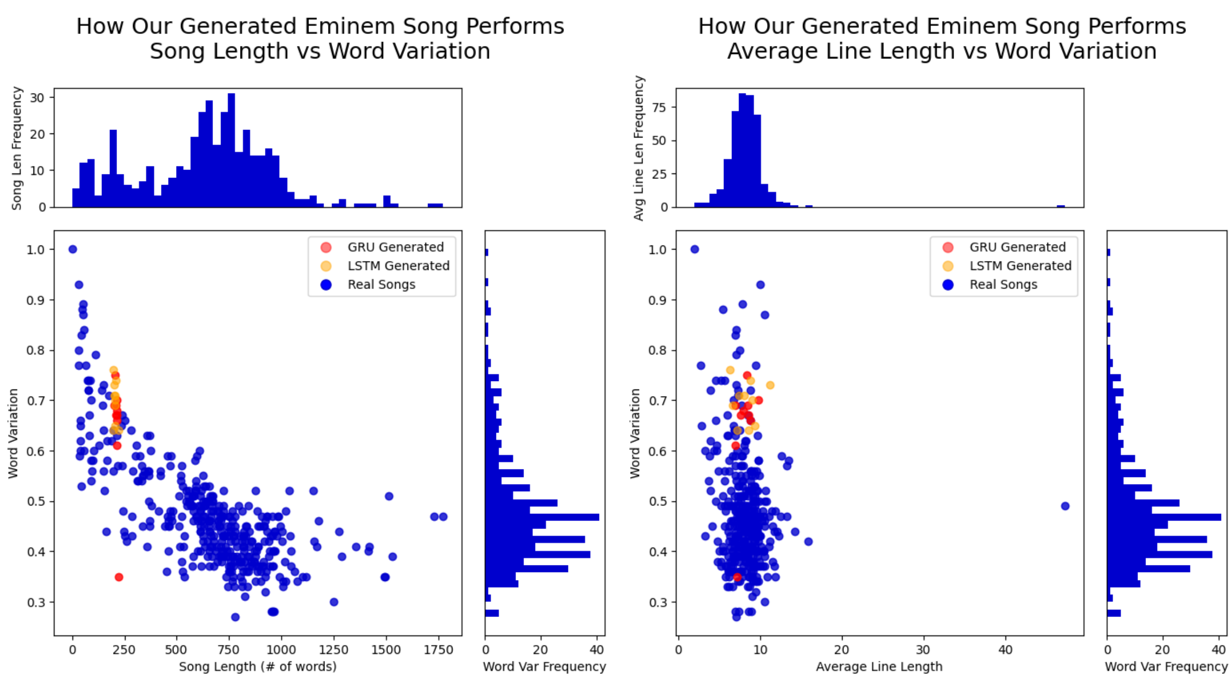

Figure 4

The song length vs word variation plot for Eminem's original songs reveals a negative correlation between song length and word variation, indicating that as songs become longer, the variety of words used decreases. Our generated lyrics also follow this pattern, showing a limited range of word variation between 0.6 to 0.8 for a fixed song length.

The line length vs word variation plot of Eminem demonstrates that the artist Eminem has a consistent preference regarding line length, and our model has successfully captured this characteristic, closely aligning with the artist's typical line length distribution.

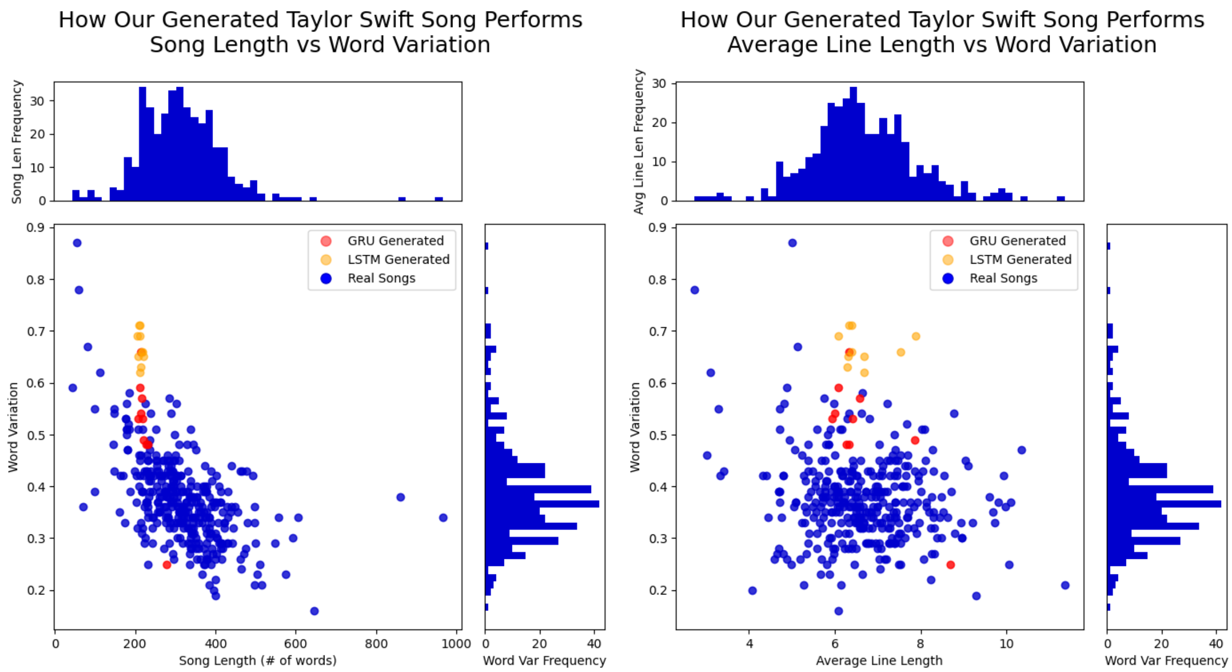

Figure 5

The song length vs word variation plot for Taylor Swift also reveals a negative correlation between song length and word variation in her original songs. Our generated lyrics follows this pattern by showing limited ranges of word variation between 0.5-0.6 and 0.6-0.7 for GRU and LSTM, respectively.

The line length vs word variation plot of Taylor Swift indicates that the artist does not have a distinct preference on line length, and our model successfully captures this characteristic as well, reflecting the absence of a specific line length pattern in the original songs of Taylor Swift.

Human evaluation:
Although the metrics discussed above revealed some interesting findings, we would like to utilize human judgment on our generated lyrics as well. Throughout human evaluation, we identified a few areas that needed improvement. One is that due to the fact that our model is character-based, the generated lyrics have some misspelled words, such as “theneself”(“themself”). We have identified this aspect as a valuable future consideration for our project. Another area is that there are some inappropriate words generated, as they are originally contained in the artists’ lyrics. The model did a good job imitating those styles, but it also raises an ethical issue, which we will discuss in further detail below.

Topic Modeling:

Choice of Models: To supplement our Generated Models, we created topic models for our selected artists to develop a better understanding of the themes, topics, and word choices that are prone to the artists we try to mimic through our generated lyrics. For each artist, we created topic models using two methods, Latent Dirichlet Allocation (LDA) and Negative Matrix Factorization (NMF). LDA is a generative probabilistic model that represents each topic generated as a collection of topic probabilities. NMF is a decomposition, non-probabilistic model that applies matrix factorization, squashing a collection of words into a finite number of topics.

Data Preprocessing & Methods: For our topic models, we further preprocessed our lyrics data. Unlike our generated models, we do not care about the structure of sentences or the speech patterns of our artists; therefore, we removed stop words and a few words we considered ad libs (i.e. “ooh” or “yeah”). We lemmatized our words, merged frequently used consecutive words into bigrams, and removed words that had fewer than 3 characters. We then further removed words that exist in over 40% of all songs belonging to the artist and words that occurred fewer than three times across all songs as we believe these words contribute little to our topics. Rather than using bag-of-words, we created our corpus using Term Frequency – Inverse Document Frequency, shortly referred to as tf-idf.

To bring out the most of each artist’s topics, we built multiple LDA topic models using 2 to 10 topic counts and measured average coherence scores for each LDA model. For our coherence computation, we used CV as it is a popular metric whose values are always positive, making it more easily interpretable. CV uses a combination of normalized pointwise mutual information and a sliding window to evaluate the cooccurrences of words. When computing our coherence scores, we only considered the top 10 words of each topic as we believe words outside this range contribute little to the comprehension of the topics. Once we recorded the average coherence scores for each model, we selected the number of topics with the highest average coherence score and built a finalized LDA model. We repeat this process for our NMF models.

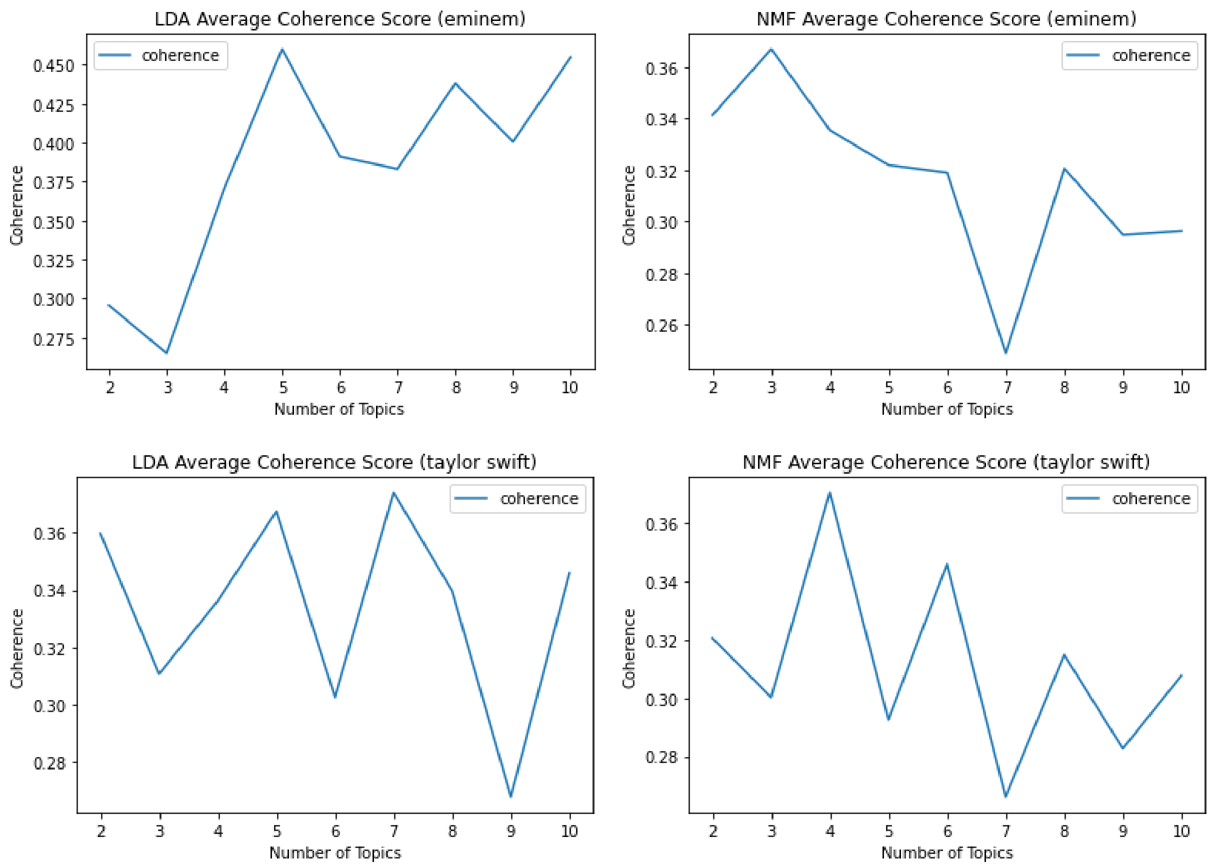

Figure 6

As we can see from Figure 6, the best performing model for Eminem is LDA. Our LDA model for Eminem was able to achieve a coherence score of 0.46 with 5 topics. This model was capable of achieving scores close to 0.5 with a larger topic count (0.49 at 18 topics); however, for the simplicity of visualization and our topic interpretations, we set max topics to 10. On the other hand, the NMF model peaks at a lower topic number. The highest coherence score the NMF model was able to achieve was 0.37 with 3 topics. Similarly, LDA performed better than NMF when we applied the same method for Taylor Swift. LDA for Taylor Swift peaked at 7 topics with a score of 0.374 and NMF peaked at 4 topics with a score of 0.371. Although the magnitude of performance difference (0.003) may seem insignificant for Taylor Swift, LDA was capable of achieving over 0.4 at higher topic count (0.42 at 14 topics). We have observed the performance of NMF models for both artists at larger topic counts; however, scores decline further, proving that a lower topic count is better for NMF.

Since most artists have many topics, we thought it would be cumbersome to plot the words of each topic. We mapped each song to the most representative topic and took the most dominant topic (topic with the most songs). Although the most dominant topic may not have the highest coherence score relative to the other topics, it reflects most of the artist’s songs; and therefore, is the most popular topic to that artist. Upon observation, we discovered that the dominant topic is often the less coherent topic (having lower coherence score than other topics); however, the words depicted by the topic can be interpreted as some of the artist’s favorite words.

We take the top 30 words of the most dominant topic generated by LDA and NMF to create word clouds for Taylor Swift and Eminem.

Figure 7

Top words for Taylor Swift's LDA model are "baby", "want", and "come". Other words that occur with less weight are "think", "tell", and "remember".Top words for the NMF model are "come", "want", and "little". Other words that occur in the topic generated by this model are "call", "baby", and "sweet". All in all, there are many words that overlap between the dominant topics generated by both models. The primary difference between the two is the weights of the words the model considers more important. If we were to summarize our interpretation of these topics, we would say the theme/topic would be desires; or perhaps, desire for love as hinted by the less important words.

Figure 8

Similarly to what we witnessed with Taylor Swift, the dominant topic generated by NMF and LDA displays many overlapping words for Eminem. Both models put high weight on words like “love”, “aint”, and “cuz”. Words that carry less weight are what differentiates the results of both models. In the topics generated by LDA, Eminem makes many references to himself. Words that occur in this model include “slim”, “shady”, and “eminem” which are his aliases. Unique words that occur in NMF’s dominant topic are “don’t”, “hate”, “cold”, and “together,” suggesting a theme that describes emotions. We summarize the words generated by this model as emotions and potentially, unrequited love.

Results: All in all, the dominant topics generated by our two models share many high-weighted words. These words are often words that should have been stop words (see future considerations section as to why we did not remove them) or words that are overly frequent across the corpus. Words with less weight are what differentiates the interpretation of the two topics. In addition, LDA performs best at a higher topic count and NMF performs best at a lower topic count. Using coherence (CV) as an evaluation metric indicates that LDA is the better performing model; however, it is difficult for us to conclude which model generates a more human interpretable topic; therefore, we leave that for you, the readers, to decide. Performances aside, we found that NMF is less computationally expensive than LDA. Our NMF models took less than half the time it took LDA to complete the training pipeline. LDA took 29.5 seconds to train 9 models for Taylor Swift using 2 to 10 topic counts, whereas NMF took 10.4 seconds.

Ethical Consideration:

Artists often incorporate strong language in their work to express emotions and represent their own style. However, we hope that our users can understand the context and sentiment of the lyrics without being exposed to explicit and offensive language. We tried to strike a balance between preserving the essence of the artists' original lyrics and ensuring the content remains suitable for a wider audience in terms of age, culture and different degrees of sensitivity.

Thus, we masked inappropriate words in our training set, so those words in our generated lyrics are masked as well. We compiled a list of profanity words based on our own knowledge of hip hop profanity, which we partially obscured with asterisks, leaving only the first character visible in our generated lyrics. By masking rather than eliminating these words entirely, we preserve the emotion and sentiment in the artists' original lyrics.

We have considered as many harmful words we could think of for our Censor List; however, we would like to disclaim that there may be words we have failed to consider. We do not condone the use of any profane language that might offend or otherwise harm our readers.

Findings and Conclusions:

Our generative model demonstrates the broader family of RNN models' capabilities in learning the patterns and styles of the artists. It illustrates the potential of these models in creative applications like generating artistic content. There are clearly certain areas where the model could be improved, namely creativity, coherence, grammar and song structure. However, these shortcomings should not obscure the true accomplishment of the models. Afterall, the tokens ingested by the NNs were not words, but individual characters. As a result, the models had to first agglomerate sequences of characters into words before they could construct the final phrases. The fact that a vast majority of the generated words actually exist in the English language is itself an impressive illustration of the capability of these models. The results can be expected to improve markedly given more data, processing power, and time.

Furthermore, we also learned that the LDA topic model performs better than NMF when it comes to CV coherence scores. The coherence score of LDA improved with increases in topic count and NMF scores do best with lower topic counts. The drawback of LDA is that it is more computationally expensive than NMF, taking more than twice as long to generate the model. As for human interpretability, it is difficult for us to see which model generates more coherent topics. Based on our own observations, we concluded that the LDA topics appeared to be more relational whereas NMF topics were more similar.

In general, we can conclude that song lyrics, like so many other phenomena currently under study, are easier to reproduce than they are to understand. While we have been able to build a model capable of rudimentary creative speech, we have made disappointingly less progress in our analysis into the essence of that speech. We can only draw encouragement from the observation that, in human development, mimicry is often the first step in a process that leads ultimately to understanding, and eventually mastery.

Future Considerations:

As mentioned previously, the generated lyrics composed by our models introduce a few misspelled words. This is due to our generative models being character based. In the future, we would like to implement a function to correct these misspelled words. We have thought of creating a dictionary carrying unique vocabulary used in an artist's song corpus. We would then iterate through each word in the generated lyrics, finding words that do not match any in the dictionary. We would then use an edit distance algorithm to find a word from the dictionary that best matches the misspelled word and replace such word.

Using Regex, we were able to identify song titles that feature other artists. Upon further exploration, we learned that some songs that did have features did not indicate such in the song title; therefore, our cleaning method was not able to capture all songs with featured artists. We would like to improve our cleaning methods by applying a similar search function to the lyrics in the hope that features would be captured via song annotations.
Although we applied stopwords to our topic models, we were unable to capture some colloquialisms as they were not in our default stopword dictionary. We have made attempts to append some of these words to our stopword dictionary; however, we were unable to include all as it was inefficient to scan through each song for unique stopwords. For instance, Eminem's most dominant topic includes words like "aint" which we consider a stopword. We hope to find a solution in the future.

Although our efforts to obtain song years have yet to be fruitful, we hope to obtain a better dataset in the future and proceed with testing of the original hypothesis. To do this correctly will require careful thought and precise definitions. For example, since the focus is on the lyrics themselves, it might be better to capture a date of copyright, instead of the release date of a recorded track. The Library of Congress may be a better source of this information than any of the various music streaming service APIs. Also, this analysis might be better executed on a carefully chosen subsample of American music, perhaps focusing on a single genre or subgenre. Lastly, in order to better differentiate the lyrics, it might be necessary to choose a custom list of stopwords specifically tuned to the peculiarities of song lyrics. Most existing lists were assembled with longform prose in mind. In retrospect, it would have been difficult to adequately perform all of these steps in the short time allotted for our project. We look forward to revising the study and carrying it through to a more satisfying conclusion after completing the MADS program.

Appendix:

1. **GPT-4 Polished Lyrics**

   With the advent of large language models, we wanted to see how the more prevalent GPT-4 can polish our generated lyrics.

   We prompted ChatGPT-4 to correct the misspellings and grammar errors, and to improve the coherence in one of the generated lyrics with Frank Sinatra's style. Comparing the two versions of lyrics below, we can see that GPT-4 can significantly improve coherence, enhance creativity and structure, and better adapt to artists' styles while refining grammar and syntax. In the future we may consider using GPT-4 to polish our generated lyrics once we have developer access to its API.

   RNN-GRU Generated Song:

   Where the music never stops,
   and then a fish where you are
   i think of you
   i think of you
   i think of you
   i thought at last i found you
   again in my love
   for you
   i love mysee a rill ridg
   come in on a wing and looking for sadnes
   and i guess what you go away, if you take me with all the dreams is real
   your hidden persaination is now to say
   that it was mine, won't look back
   my heart says it's done
   it's over, it's over, it's over, but when i see the looring, get toice aser sang
   love to tell me at you, in my heart.
   i those earo, that need my special cale,
   it's a lifetime of white as her lips of more than just a sang like stars with oh a fight him
   glow on a winner
   but i believely
   and me, tell.
   i leave yourself on alp ourds floating on a rove.
   is there a while,
   will you have a rocart and found it free,
   and in this will we leave feel so
   nothing i move.
   my schemes, my dream on the fireless hadd
   never feel a lettle in your eyes, is there no more
   so darling
   where the stars in the pocket, or ride a rocket, price our me, people will say we're in love.

ChatGPT-4:
Verse 1:
Where the music never stops,
A fish swims where you are,
I think of you, I think of you,
I thought at last I found you,
Again in my love for you,
I love myself, a real ridge,
Coming in on a wing, looking for sadness.
Chorus:
And I guess you'll go away,
If you take me with all the dreams that are real,
Your hidden persuasion is now to say,
That it was mine, won't look back,
My heart says it's done,
It's over, it's over, it's over.
Verse 2:
When I see the longing, love to tell,
Me and you, in my heart,
Heroes that need my special care,
A lifetime of white, as her lips,
More than just a song like stars,
A fight, a glow, a winner,
But I believe, and me, I'll tell.
Bridge:
I'll leave you on all our floating roads,
Will you have a rocket and find it free?
In this, will we leave and feel so,
Nothing will move,
My schemes, my dreams, in the fireless hand,
Never feel a little in your eyes,
Is there no more?
Chorus:
And I guess you'll go away,
If you take me with all the dreams that are real,
Your hidden persuasion is now to say,
That it was mine, won't look back,
My heart says it's done,
It's over, it's over, it's over.
Outro:
So, darling,

Where the stars in the pocket or ride a rocket,
Price our love, people will say we're in love.

2. **Attempting to extract year labels**
   Before changing direction, we made 4 major attempts to obtain useful year labels:
   - **MusicBrainz database**: This crowdsourced database is mentioned in several projects, including some significant academic studies. It is crowdsourced, but appears to be comprehensive. At first we attempted to download the necessary portions of the database as flat files, but the database schema was heavily normalized, poorly documented, and more complex than expected. It was during examination of the underlying schema that we began to realize the breadth of the myriad complications involved in extracting a single definitive year associated with a given song. There are several possible candidates, including release dates, recording dates, and attribution dates to name a few. In the end, we focused on release dates and pulled in the minimum such date we could find for each artist / title combination. While we realized the potential flaws with this approach, we hoped the scope of any confounding factors would be minimal. We decided to use the API instead of the database files to avoid reconstructing the complex database relationships. Due to rate limits, we had to pare down the dataset significantly. After conducting a manual spotcheck of the results, we concluded that the dates returned were not trustworthy

   - **Spotify API**: Concluding that the bad dates from MusicBrainz might be a simple matter of data integrity, we hoped for better luck using a curated commercial data source. A script was written to loop through our songs and pass their titles and artists to the Spotify API, carefully timing the calls so as to avoid rate limit violations. The script ran for close to 24 hours and terminated normally. When we reviewed the years returned, we found the same issues as were found in the MusicBrainz labels

   - **Wikipedia**: In a final effort to retrieve meaningful dates, we turned to Wikipedia. The site has quite a comprehensive collection of popular songs, along with the associated artists and original release dates. A survey of this collection revealed a lack of standardization in page format, but we were able to identify a handful of formats that seemed to recur for most songs. A script was built with 2 parts. The first part gathered the page URLs for every song that had a matching title and artist in our dataset. This part took several hours, but ran successfully and returned matches for about 68% of the songs in our dataset. The second part was designed to visit each URL and check various page elements for a release date. The script seemed to work well, but after a couple hours of running, it would crash with an unintelligible error. We believe the error was due to Wikipedia's equivalent of a rate limit. After several attempts, the effort was abandoned.

- **Imputation**: In a final effort to extract at least some value from our previously harvested data labels, we attempted to isolate and impute extreme year values for each artist. This approach relied on our assumption, borne out by cursory observation, that most of the years were within reasonable range of the true year of release. We thus compared each year label to the artist's mean value and separated out those samples beyond a standard deviation threshold. We experimented with several SD values including 0.87, 1.08, 1.50. Ultimately, we used the statistical IQR as the cutoff threshold. This approach, while reasonable, did not yield a significant improvement in the results. Nonetheless, we attempted to cluster the lyrics and compare them to the imputed date labels. Since the resulting labels were still in doubt, and the analysis was unsurprisingly inconclusive, we finally decided to abandon the whole enterprise.

Statement of Work:

Horner, Josh:

- Initial data exploration
- Extraction of year labels (not used)
- Clustering of Lyric Vectors
    - Comparison to Decades / Eras (not used)
    - Comparison to Genres

Jiang, Yanying:

- Generative Models
    - RNN-GRU
    - RNN-LSTM
- Creating Evaluation Metrics for Generative Models
- TensorBoard Plot

Tang, Kenny:

- Data Cleaning
- Topic Modeling
    - LDA
    - NMF
- Word Clouds
- Scatterplot for Generative model

Team:

- Paper
- Poster
- Exploratory Analysis
- Github
- Data Quality

Github:

https://github.com/kennyta0204/Capstone-Project

Acknowledgements:

We would like to thank Michelle LeBlanc, Dr. Elle O'Brien, and Rachael Tatman for their support, feedback, and guidance.

References:

Antoniou, M. (2018, June 27). Text analytics & topic modeling on music genres song lyrics. Towards Data Science. https://towardsdatascience.com/text-analytics-topic-modelling-on-music-genres-song-lyrics-deb82c86caa2

CHENXIN. (n.d.). Guitar Silhouette Vector Material Silhouette PNG Transparent [Image]. pngtree. https://pngtree.com/freepng/guitar-silhouette-vector-material_6224232.html

Gill, H., Lee, D., & Marwell, N. (2020). Deep Learning in Musical Lyric Generation: An LSTM-Based Approach (Vol 1, Issue 1). The Yale Undergraduate Research Journal. https://elischolar.library.yale.edu/cgi/viewcontent.cgi?article=1000&context=yurj

Kapadia, S. (2019, August 19). Evaluate topic models: Latent Dirichlet allocation (LDA). Towards Data Science. https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

MusicBrainz API - MusicBrainz. (n.d.). Retrieved March 20, 2023, from https://musicbrainz.org/doc/MusicBrainz_API

Neisse, A. (2022). Song lyrics from 79 musical genres. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres?select=artists-data.csv

Text generation with an RNN. (2023, March 8). TensorFlow. https://www.tensorflow.org/text/tutorials/text_generation