UNIVERSITY OF CALIFORNIA

Los Angeles

Application of Machine Learning Model in Generating Song Lyrics

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Wesley Lau

2021

ABSTRACT OF THE THESIS

Application of Machine Learning Model in Generating Song Lyrics

by

Wesley Lau
Master of Applied Statistics
University of California, Los Angeles, 2021
Professor Yingnian Wu, Chair

(Abstract omitted for brevity)

The thesis of Wesley Lau is approved.

To my loving parents who have supported me in all things

Special mention to my friends and collaborators Jen and Andrew

And last but not least the encouraging teaching faculty at UCLA

Thank you

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

(Acknowledgments omitted for brevity.)

CHAPTER 1

Introduction

More than 150 years ago, humans were debating whether algorithms or machines could produce original content. Ada Lovelace, who is credited to be for creating the world's first computer program alongside Charles Babbage, claimed the computer program had no power to author anything original and could only perform what was ordered of it when speaking of the analytical machine. Although this hypothesis has been contested in the many years since her letters have been published, many people still think of art and music as a domain dominated by the human brain as it requires a degree of creative skill.[1]

Much has been written about the creativity of humans. There is an abundance of works of arts that are thought of as only achievable through human ingenuity. As Albert Einstein wrote, "The true sign of intelligence is not knowledge but imagination." Most people wouldn't consider modern day computers to have imaginations or a even an awareness of the mind. These things are seemingly incomprehensible for computers and do not seem possible to produce through deterministic algorithms. However, this has not stopped programmers and researchers from creating machine learning algorithms that surprise its authors with the results they generate. The programmer is unable trace back how the algorithm produced its results from this process. Considering this, it might be time to re-evaluate what role computers can play in the creative space.

In this paper we will examine what role machine learning can play in the creation of new artistic and creative content, specifically song lyrics. I will include the methodology and data preparation I did to form the training dataset of song lyrics. After data cleanup, I will examine the songs lyrics and artists on a couple of features including musical genre, song length, number of songs written, repetition score, and Term Frequency–Inverse Document Frequency (TF-IDF) vectors. Finally, I will generate novel song lyrics from the GPT-2 model software package using a training subset of data. We can examine the product of the machine generated song lyrics and see how it compares to actual song lyrics.

CHAPTER 2

Background and Motivation

2.1 Prior Work

2.1.1 Emmy

There is a history of using pattern recognition and machines to learn the rules of music and create something original. Composer David Cope was an early adopter of computer assisted creation of music. He wrote an algorithm called Experiments in Musical Intelligence (EMI) that helped him complete the opera called ”Cradle Falling,” which ended up being the best reviewed opera of his career two years later.

Motivated by his success, Cope went on to modify the algorithm, later renamed Emmy, to compose pieces of music that imitated the style of the renowned composer Johann Sebastian Bach. Cope was able to create a database out of the catalogue of compositions written by Bach. From there, he extracted certain segments corresponding to motif's or recurring sequences of notes in the body of work. In his analysis of different composers like Brahms, Chopin, Mozart, and Bach, each composer had particular patterns of notes that they were strongly drawn to.

After breaking the composition database down into fragments, Cope developed an algorithm for arranging those fragments according to a predetermined musical structure. He thought that this structure would lead to resonate emotion and lead to musical tension in the piece. The top level instructions were able to guide the machine to create new compositions from Bach's music fragments by linking them together following certain pre-made rules. This use of algorithms to mimic Bach's compositions was certainly not the last attempt to use machines to aid in the creation of music.[1]

2.1.2 The Flow Machine

François Pachet, a composer and researcher that worked for Sony Computer Science Laboratory in Paris as well as Spotify, has the accomplishment of creating the first AI Jazz improviser. Pachet used Markov Chains to create the music and lyrics. In this model of machine aided creation, the author needs to provide structure to the song, and the computer will fill in the notes or words. The constraints imposed on the generated music can range from using all the notes in the chromatic scale to capturing the style of Bob Dylan's lyrics.

After the rules were created, the Flow Machine algorithm filled in the rest of the structure with its choice of notes or words based on the Markov Model probabilities. Critics of machine generated compositions said the memory-less property of Markov Models meant the composition lost long term structure and could not form coherent stories with words.[1]

2.1.3 GPT-2 and the Transformer Model

In contrast to Markov Chains (also called Markov Models), the GPT-2 toolkit doesn't require any structures to generate, only training data. It uses a transformer model.

Transformers are a machine learning structure with a long term memory that in theory could bridge the gap and make coherent pieces. I wanted to see if it was possible to use GPT2, an open-source transformer toolkit, to generate music lyrics that would be both understandable and have a common structure.

Almost all language models parameterize the words of a language into vectors. The software package 'word2vec', for example, embeds tokens or word-stems into a high-dimension vector space. This allows the model to represent the semantic meaning of words as vectors, which are easier for computers to store and manipulate. This type semantic embedding is

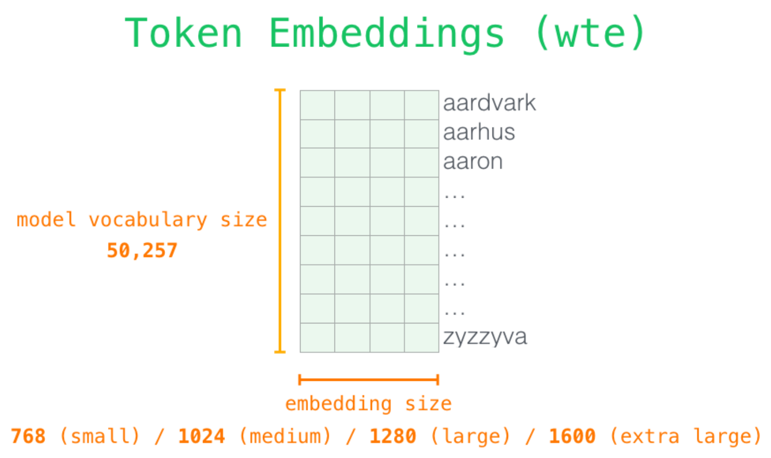


Figure 2.1: Example of Word2Vec Embedding

useful for predicting the next word given a chunk of preceding words.[2] See Figure 2.1 below for an example of how the dictionary of English words (50,257 words long) might be encoded into a collection of vectors.

As it turns out, this method is also useful to form the building block of the decoder, a fundamental part of transformer architecture. The model that we use from OpenAI, called GPT-2, uses a chain of these decoder blocks to create its transformer model. One important fact is this model is autoregressive in nature, meaning that after each word is generated, it gets added to the block of text that is input to the model in the next step of generation. It is also unidirectional, meaning it does not modify the embedding the first word after looking at the second.

The specifics of how attention is implemented is described in the journal paper titled "Attention is All You Need" by Vaswani et. all [3]. We can picture the attention mechanism as as a way to build in memory of the previous words in the passage. "It bakes in the model's understanding of relevant and associated words that explain the context of a certain word before processing that word (passing it through a neural network)." This is accomplished by scoring previous words on their relevance to the current word, combined with using their

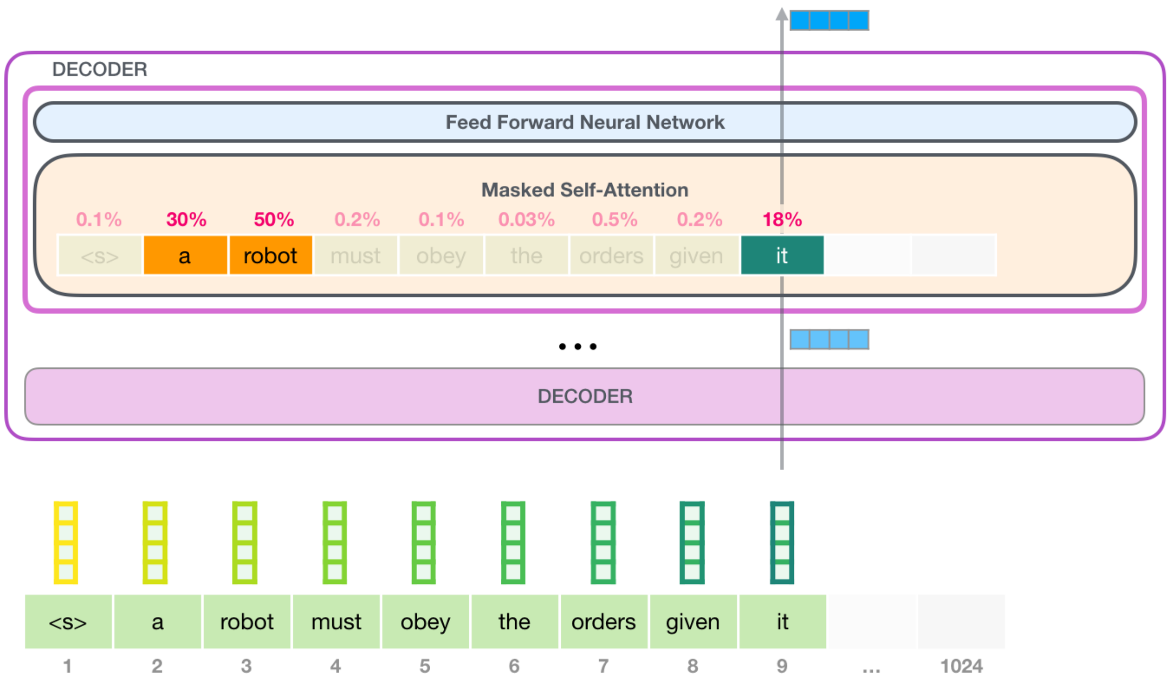


Figure 2.2: Example of GPT-2 Self-Attention

vector word embeddings. [2]

Computers have a hard time understanding sentences that include many different contextual references. Consider following quotation taken from Isaac Asimov's science fiction story *I, Robot*. "**Second Law of Robotics:** A robot must obey the orders given it by human beings except where such orders would conflict with the First Law." As we can see from the illustration in Figure 2.2, the understanding of "it" in that line is informed by the words "a" and "robot" earlier in the sentence. The input words will go through many of these decoder blocks and will eventually produce an output vector. This output vector is multiplied by a pre-built word embedding dictionary (around 50k words) that will give us the vector of probability scores for the next word, i.e. how likely each word will be the next word in the sequence. We now have the choice to choose the word with the highest probability, or sample from a list of words with the probability associated to its score.[4]

Even with all these limitations, I set out to show that this model can produce fully formed song lyrics that can have a consistent topic and are fairly comprehensible from start to finish. I wanted to show that artistic and creative activities traditionally dominated by humans can be supplemented by the computer assistance.

2.2 Research Question

I will analyze of the song lyrics of a collection English music. I also will explore the use of GPT-2 transformer architecture in aiding the creation of novel song lyrics. The properties of the generated text will be compared quantitatively and qualitatively with the original dataset.

CHAPTER 3

Methodology

3.1 Dataset

My final dataset is a conglomeration of three different sources. As shown in Figure 3.1, the first source of data is song lyrics dataset obtained online that was originally scraped from the website AZlyrics. From this table, the important columns are the song name, artist name, and the full text of the song. This body of text is already sufficient to start utilizing the machine learning GPT-2 toolkit to create new lyrics by training on the existing song lyrics. However, I also wanted to investigate if it was possible to create song lyrics of a particular genre or style of music. Furthermore, I wanted to analyze different lyrics based on the categorical variable of genre. Since the original dataset did not include the song genre column, I had to augment that data with two other sources. One of the sources was an API called "theAudioDB", which included an endpoint that returned the artist genre from input of the artist name. From this dataset, we were able to label the artists and ultimately individual tracks with a single genre tag. To get around the problem of artists with non-unique names (multiple artists or groups may have the same name), I made use of a third source called the musicbrainz dataset. This open-source collection of song's and their metadata contains a open-source standard for the unique ID for artists. Using this standardized ID, I was able to query a different endpoint of theAudioDB API and get the genre for the unique artist ID for some of songs.

In the end, my datatable contained 122289 rows of songs and the remaining columns:

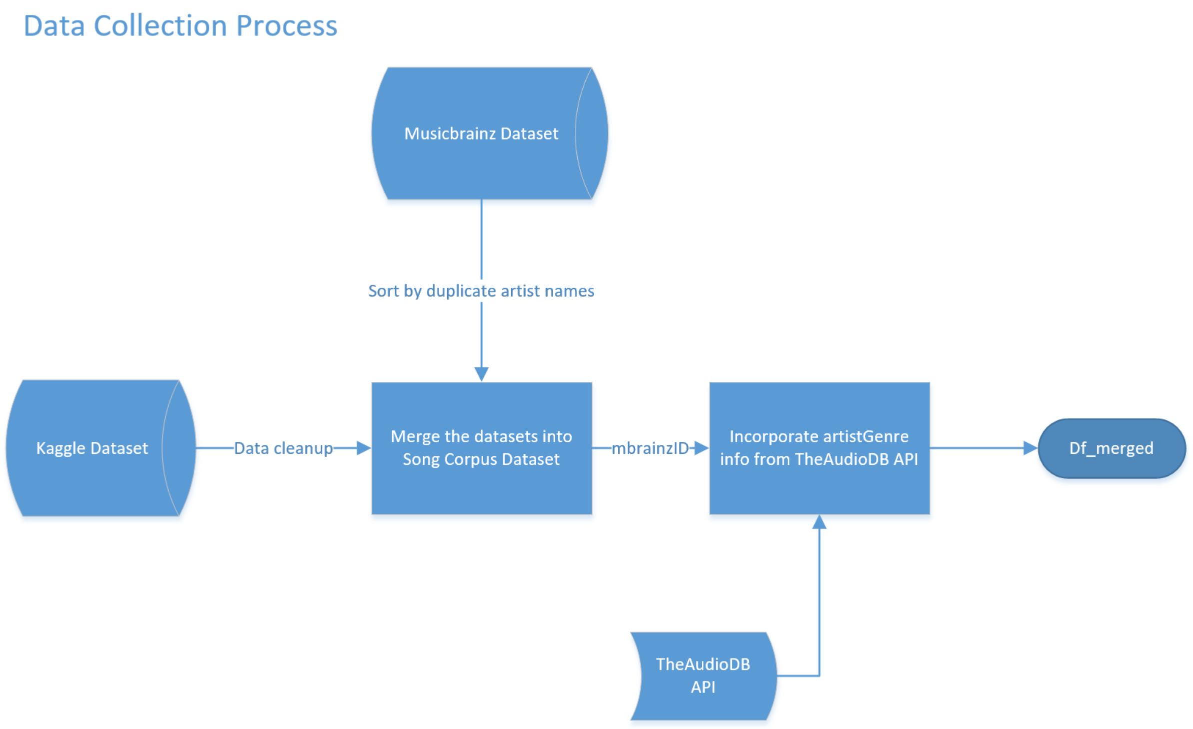

Data Collection Process
Musicbrainz Dataset
Sort by duplicate artist names
Kaggle Dataset
Data cleanup
Merge the datasets into Song Corpus Dataset
mbrainzID
Incorporate artistGenre info from TheAudioDB API
Df_merged
TheAudioDB API

Figure 3.1: Data Collection Process

"artistName", "artistURL", "text", "songName", "songURL", and "musicType".

Variable	Type	Definition
artistName	categorical	Name of the artist. A particular name may refer to multiple different entities.
artistURL	categorical	Link to the artist's page on AZlyrics website.
songName	categorical	Name of the song.
songURL	categorical	Link to the song lyrics on AZlyrics website.
musicType	categorical	Label that represents either the genre or style of the song by that particular artist. See Data Cleanup section in the Methodology Chapter for details of how its derived.
text	text	All of the lyrics of a song as posted on https://www.azlyrics.com. See Data Cleanup section in the Methodology Chapter for details on the post-processing done.

Table 3.1: Dataset Variables

As you can see from the Table 3.1, the only variables that involved some text post-processing or are transformed from their original state are the "text" and "musicType" columns. The other variables come directly from scraping the AZlyrics website. User Albert Suarez implemented a software package called azlyrics-scraper on python that is freely available on his Github repositiory. His Github page, https://github.com/AlbertSuarez/azlyrics-scraper, also links to a cloud storage repository where the results of the webscraping are uploaded to periodically.

3.2 Data Cleanup

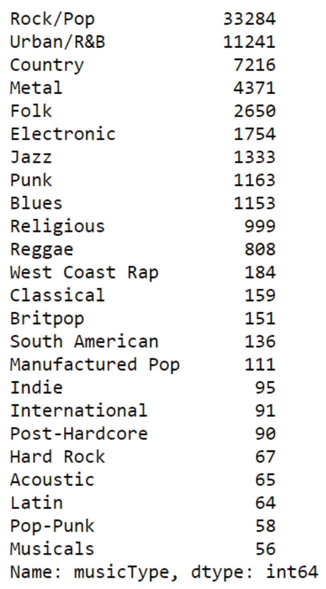
The 'text' data field is derived from the original song lyrics on AZlyrics website. Text post-processing was done so that the lyrics only represented words that were spoken in the song. Any words in italics, parenthesis, or brackets were removed. This usually represented things like extra song information or signalled a repetition of the chorus lyrics. For example "(live unplugged)" or "chorus 2x". I also removed any lyrics that were from duplicate songs. This includes duplicate songs by the same artist and duplicate song text attributed to different artists. This could be because a particular song featured multiple artists so it might be attributed to all of them.

Finally, I removed any songs lyrics that were not written in English, since I wanted to generate only songs written in English. To do this I used the "detect_langs" function from the langdetect library in python to return a probability score that the lyrics were written in English and used a score cutoff to keep those that were above 50%.

MusicType is a variable derived from the two different endpoints of theAudioDB API database. One method I used searched for genre information using only the artistName as text input. A second method used theAudioDB endpoint to find an artist's songs using the artistID variable associated with that entity. The variable artistID is derived from the MBID field in the MusicBrainz dataset. Per the MusicBrainz project documentation, MusicbrainzID (MBID) is a unique identifier that is given to each artist in their database.[5]

One thing we encountered when searching for artists through their name as text was that there were multiple artists sharing the same name. To get around this issue of ambiguous aliasing, the MBID field was used to make use of the property that it was a unique identifier for the artists. I was able to get genre labels assigned to a greater set of my song lyrics using this method.

The variable 'MusicType' is the combination of the returned genre or music style labels from multiple methods that ended up classifying the largest number of songs in our song



```
Rock/Pop            33284
Urban/R&B           11241
Country              7216
Metal                4371
Folk                 2650
Electronic           1754
Jazz                 1333
Punk                 1163
Blues                1153
Religious             999
Reggae                808
West Coast Rap        184
Classical             159
Britpop               151
South American        136
Manufactured Pop      111
Indie                  95
International          91
Post-Hardcore          90
Hard Rock              67
Acoustic               65
Latin                  64
Pop-Punk               58
Musicals               56
Name: musicType, dtype: int64
```

Figure 3.2: MusicType Frequency Chart

dataset. See Figure 3.1 for a flow chart of how the data sources was manipulated from its original sources. In cleaned dataset, the different music genres and their frequency count are shown below in Figure 3.2. Only genres of music that contain 50 or more song lyrics in our lyrics database are shown in Figure 3.2.

The final dataset included 122288 rows of cleaned song lyrics. Reference Table 3.1 for the variables in my dataset after the data cleanup stage. Three more variables were derived from these columns to explore the data in the next section. These are the tokens, stems, and n_stems variables, which are all derived from the text of the song lyrics. Furthermore, I removed outliers in the dataset in terms of songs that were much longer than the others (one of them aptly named '30,000 Word Rap Song!'). After investigation, I kept the songs that had a small number of stems since it seems to come from songs with very repetitive lyrics or choruses. Instead of duplicating the text of the chorus again and again, the convention at AZlyrics.com is to put text like 'chorus 2x' within the lyrics or '(repeat chorus)' in place of the repeated lines. Even if the total song time was average, this would result in a low number of words per song after data pre-processing.

CHAPTER 4

Experiments

4.1 Exploratory Analysis

I plotted a bar chart showing the most prolific artists in terms of number of songs for several musicTypes. Figure 4.1 shows the top 40 Rock/Pop style artists in my final dataset that had the most songs collected by AZlyrics. As we can see, the most prolific Rock/Pop artist on the AZlyrics database is Barbara Streisand, with around 400 song lyrics. There are 1032 unique artists in the Rock/Pop style, 40 of which are included in this figure.

Interestingly, there aren't many artists shown that I would consider modern or contemporary artists. Britney Spears, appearing the 15th position in this chart, is a pop star popular in the 1990's. U2 is an prolific rock band that released their debut studio album even before that in 1980. At first glance the artists or pop groups most represented on AZlyrics website seem to be ones that began their careers before the year 2000 or so. Two hypothesis as to why this is come to mind. One is that more contemporary artists work for music labels that are more protective of their intellectual property. Their business label might spend more time and resources to take the artist's song lyrics off from the AZlyrics website. The other theory is that popular artists that have endured until today simply had more time to write songs in their longer career, and that has been captured in our dataset.

Regardless, this may result in the training data being biased towards older song lyrics rather than contemporary. I don't predict that should cause any issues with lyric generation. Art and music takes inspiration from what was done previously. There should be similarities

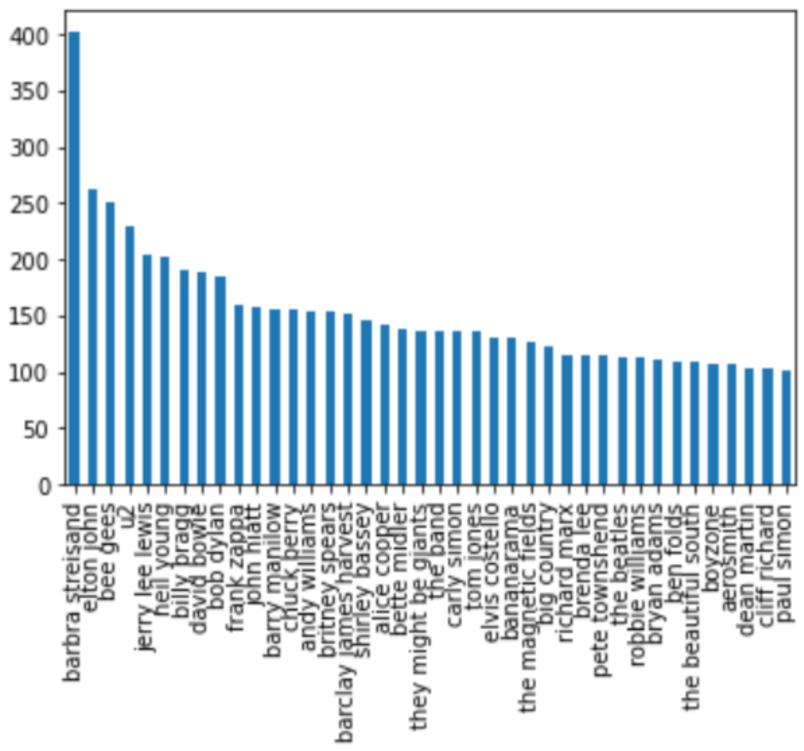


Figure 4.1: Top 30 Rock/Pop Artists

between pop and rock song lyrics from the 1970's and the 1980's and those created afterwards.

Figure 4.2 shows the top 15 most prolific artists Rock/Pop style artists in my final dataset. In this case, the artist E-40 stands out as having many more songs collected by AZlyrics than the rest. He has about 500 songs listed. The next highest artists Chris Brown and Wiz Khalifa have 222 and 200 songs respectively, less than half of that of E-40. This could be due to the frequent number of collaborative albums and featured singles or guest appearances done by E-40.

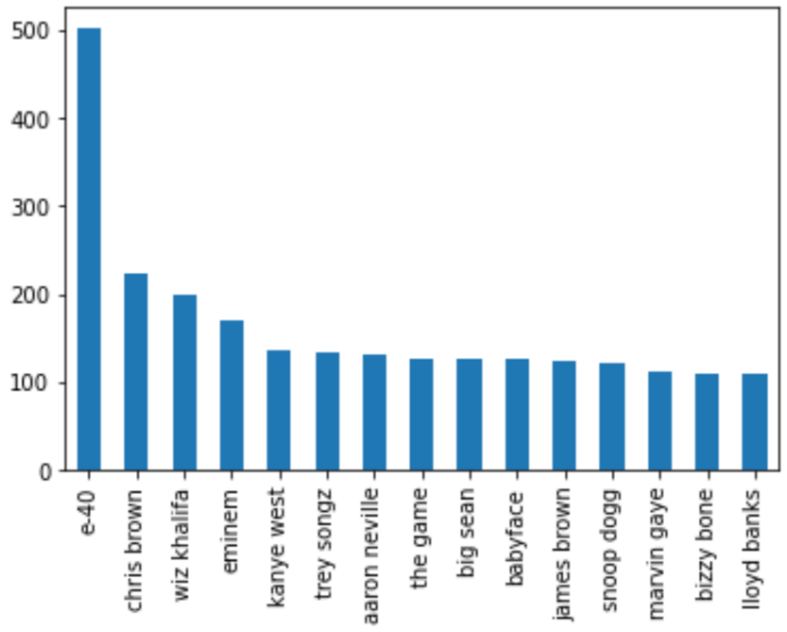


Figure 4.2: Top 15 Urban or R&B Artists

There are 279 unique artists in the Urban/RB style, much fewer than that of Rock/Pop. However, the artists that appear on this list look much more contemporary and recognizable. I have heard music from seven out of the top ten most prolific artists in Figure 4.2.

Again we see a similar pattern in Figure 4.3, as the most prolific star Johnny Cash distinguishes himself amongst other artists. This chart shows the 15 most prolific artists in the country style. He authored or coauthored more than 300 songs on the AZlyrics database.

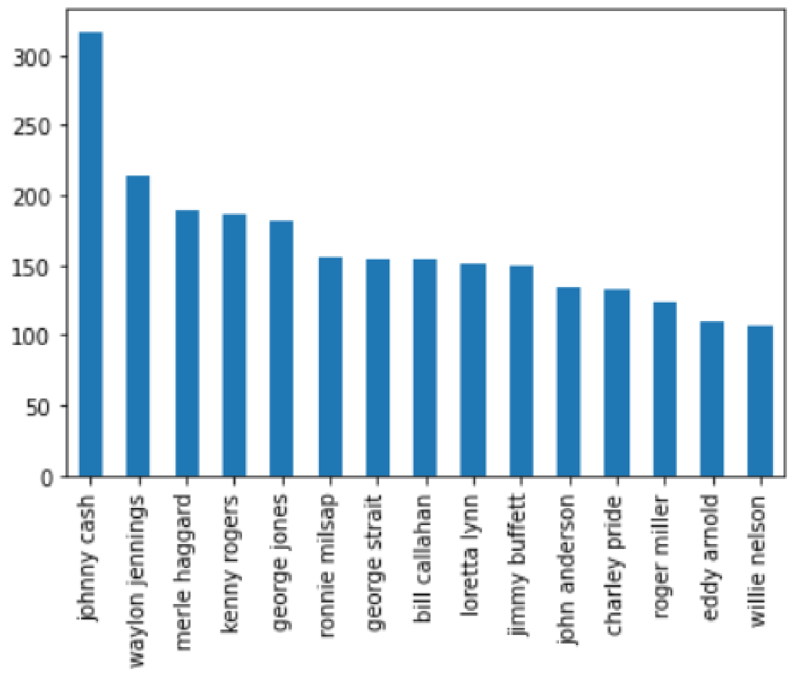


Figure 4.3: Top 15 Country Artists

His output is unmatched, whereas the rest of the list seem to have a similar number of songs.

There are only 123 unique artists in this style of music in our collection of song lyrics. This amounts to only 7216 songs classified as this musicType. As we shall see in the song generation section, this is not enough to train our GPT-2 algorithm to generate coherent song lyrics. We will have to use alternative methods in order to create a convincing mimicry of country music lyrics

I wondered how many words do song lyrics typically have? In order to answer this, I plotted a histogram of word counts for the songs in our dataset. Figure 4.4 shows the number of stems

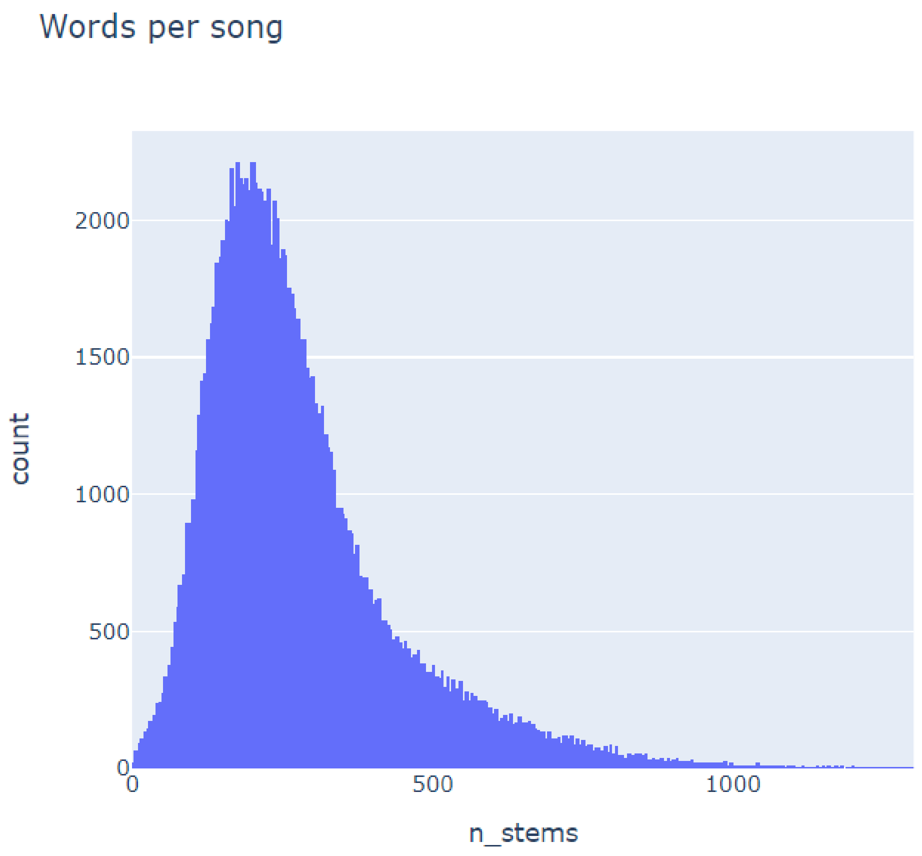


Figure 4.4: MusicType Frequency Chart

4.2 Text Analysis using Term Frequency–Inverse Document Frequency (TF-IDF)

One of the ways that text can be compared between different documents is to compare the similarities and differences in words used across the different documents. Words that are commonly used in the English language, like "the" or "a" may not reveal much about the nature of the document, but words that appear relatively infrequently in the whole collection of documents should give us a stronger understanding of the topic of each individual text. The purpose of using Term Frequency–Inverse Document Frequency (TF-IDF) is to find such unique terms in each of the song lyrics.

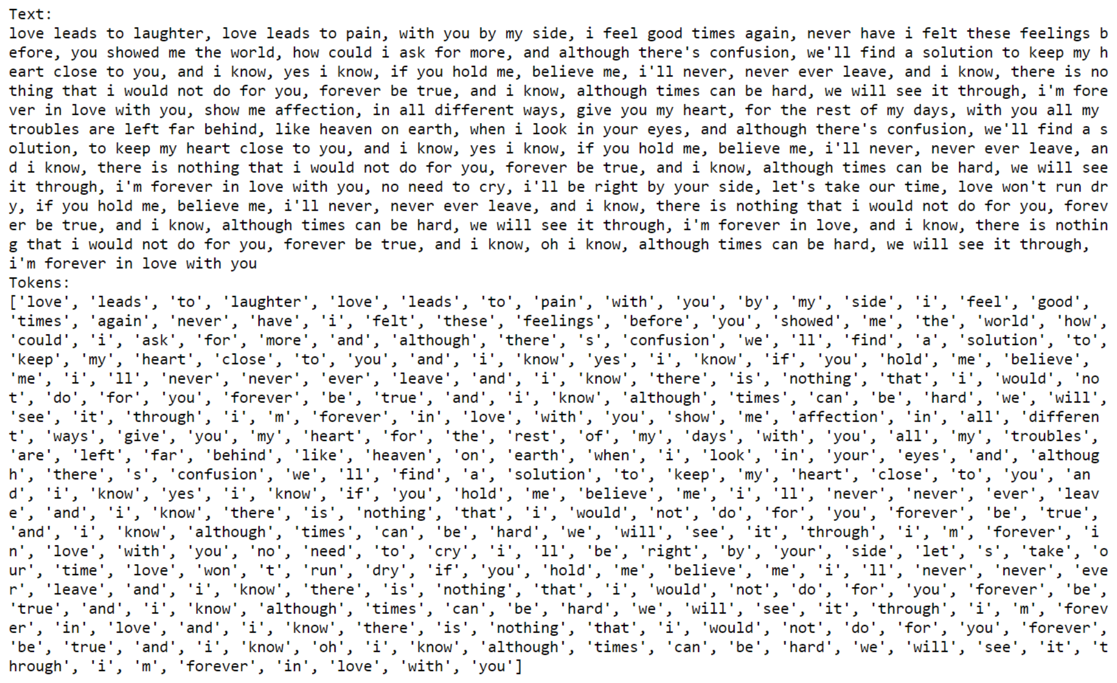
4.2.1 Word Tokens

To begin, you have to tokenize the words in the song lyrics. This is a process of picking out individual words or tokens from a whole field of text. In our case, the lyrics are tokenized into an array of words and word parts. Performing this process will get rid of any punctuation. Furthermore, fragments of words after the punctuation will remain as well. I used the python package "nltk" to tokenize the lyrics of each song.

I included a sample of a set of lyrics that have been split into tokens in Figure 4.5. Notice the lack of punctuation and capitalization in the set of tokenized words. The tokens also contain repeated words or word fragments. This will be used later on to derive the variable word stems that is used in the later calculations.

4.2.2 Stemming

Stemming converts all the tokens in the present tense and in singular form. This base form is better for extracting the meaning of the words instead of their particular format. Using Stemming, we were able to reduce the number of unique tokens or words from 188317 to



```
Text:
love leads to laughter, love leads to pain, with you by my side, i feel good times again, never have i felt these feelings b
efore, you showed me the world, how could i ask for more, and although there's confusion, we'll find a solution to keep my h
eart close to you, and i know, yes i know, if you hold me, believe me, i'll never, never ever leave, and i know, there is no
thing that i would not do for you, forever be true, and i know, although times can be hard, we will see it through, i'm fore
ver in love with you, show me affection, in all different ways, give you my heart, for the rest of my days, with you all my
troubles are left far behind, like heaven on earth, when i look in your eyes, and although there's confusion, we'll find a s
olution, to keep my heart close to you, and i know, yes i know, if you hold me, believe me, i'll never, never ever leave, an
d i know, there is nothing that i would not do for you, forever be true, and i know, although times can be hard, we will see
it through, i'm forever in love with you, no need to cry, i'll be right by your side, let's take our time, love won't run dr
y, if you hold me, believe me, i'll never, never ever leave, and i know, there is nothing that i would not do for you, forev
er be true, and i know, although times can be hard, we will see it through, i'm forever in love, and i know, there is nothin
g that i would not do for you, forever be true, and i know, oh i know, although times can be hard, we will see it through,
i'm forever in love with you
Tokens:
['love', 'leads', 'to', 'laughter', 'love', 'leads', 'to', 'pain', 'with', 'you', 'by', 'my', 'side', 'i', 'feel', 'good',
'times', 'again', 'never', 'have', 'i', 'felt', 'these', 'feelings', 'before', 'you', 'showed', 'me', 'the', 'world', 'how',
'could', 'i', 'ask', 'for', 'more', 'and', 'although', 'there', 's', 'confusion', 'we', 'll', 'find', 'a', 'solution', 'to',
'keep', 'my', 'heart', 'close', 'to', 'you', 'and', 'i', 'know', 'yes', 'i', 'know', 'if', 'you', 'hold', 'me', 'believe',
'me', 'i', 'll', 'never', 'never', 'ever', 'leave', 'and', 'i', 'know', 'there', 'is', 'nothing', 'that', 'i', 'would', 'no
t', 'do', 'for', 'you', 'forever', 'be', 'true', 'and', 'i', 'know', 'although', 'times', 'can', 'be', 'hard', 'we', 'will',
'see', 'it', 'through', 'i', 'm', 'forever', 'in', 'love', 'with', 'you', 'show', 'me', 'affection', 'in', 'all', 'differen
t', 'ways', 'give', 'you', 'my', 'heart', 'for', 'the', 'rest', 'of', 'my', 'days', 'with', 'you', 'all', 'my', 'troubles',
'are', 'left', 'far', 'behind', 'like', 'heaven', 'on', 'earth', 'when', 'i', 'look', 'in', 'your', 'eyes', 'and', 'althoug
h', 'there', 's', 'confusion', 'we', 'll', 'find', 'a', 'solution', 'to', 'keep', 'my', 'heart', 'close', 'to', 'you', 'an
d', 'i', 'know', 'yes', 'i', 'know', 'if', 'you', 'hold', 'me', 'believe', 'me', 'i', 'll', 'never', 'never', 'ever', 'leav
e', 'and', 'i', 'know', 'there', 'is', 'nothing', 'that', 'i', 'would', 'not', 'do', 'for', 'you', 'forever', 'be', 'true',
'and', 'i', 'know', 'although', 'times', 'can', 'be', 'hard', 'we', 'will', 'see', 'it', 'through', 'i', 'm', 'forever', 'i
n', 'love', 'with', 'you', 'no', 'need', 'to', 'cry', 'i', 'll', 'be', 'right', 'by', 'your', 'side', 'let', 's', 'take', 'o
ur', 'time', 'love', 'won', 't', 'run', 'dry', 'if', 'you', 'hold', 'me', 'believe', 'me', 'i', 'll', 'never', 'never', 'eve
r', 'leave', 'and', 'i', 'know', 'there', 'is', 'nothing', 'that', 'i', 'would', 'not', 'do', 'for', 'you', 'forever', 'be',
'true', 'and', 'i', 'know', 'although', 'times', 'can', 'be', 'hard', 'we', 'will', 'see', 'it', 'through', 'i', 'm', 'forev
er', 'in', 'love', 'and', 'i', 'know', 'there', 'is', 'nothing', 'that', 'i', 'would', 'not', 'do', 'for', 'you', 'forever',
'be', 'true', 'and', 'i', 'know', 'oh', 'i', 'know', 'although', 'times', 'can', 'be', 'hard', 'we', 'will', 'see', 'it', 't
hrough', 'i', 'm', 'forever', 'in', 'love', 'with', 'you']
```

Figure 4.5: Example of Tokenized Lyrics

150568 stems.

4.2.3 Repetition of Words

One metric I constructed to measure the repetitiveness of words in a song is called repetition score. It is defined as a ratio of unique word stems to the total number of stems for each song. The formula for this metric is given in Equation 4.1. This gives us a ratio between 0 and 1 for every song in our dataset. A lower ratio means more words are repeated in the song lyrics, while a ratio of 1 means all words in the song are unique.

$$\text{Repetition Score} = \frac{(\text{Number of unique stems in the song})}{(\text{Total number of stems in the song})} \tag{4.1}$$

Taking a random sample of the artists in our dataset, i was able to plot boxplots of the distribution of Repetition Score over all their songs for 10 different artists. This is shown in

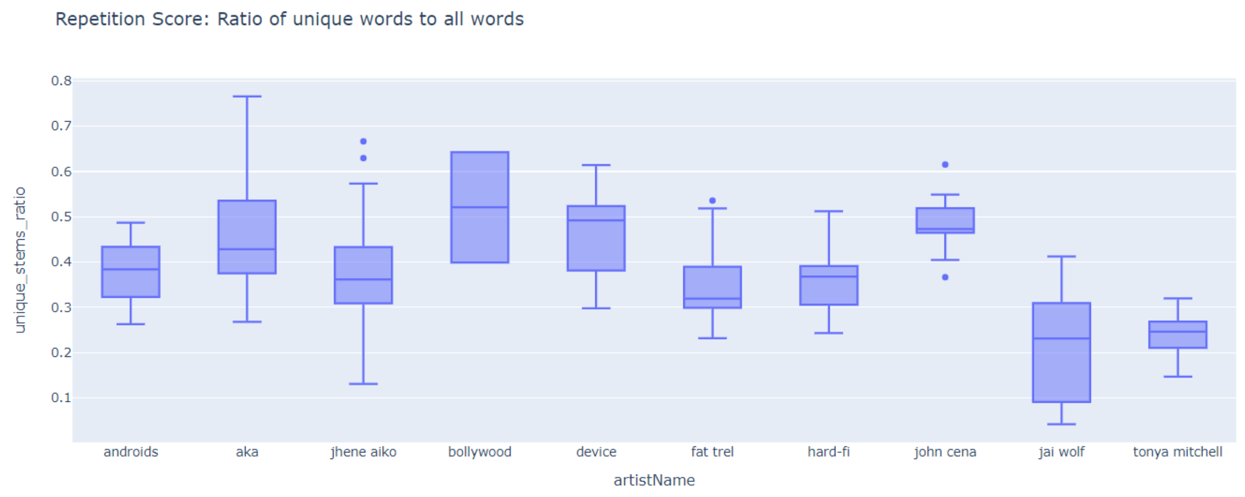


Figure 4.6: Repetition Scores for 10 Artists

Figure 4.6. Out of the 10 samples shown, the most repetitive artist seems to be Jai Wolf, the least repetitive song belongs to AKA, and the artist with the widest range of compositions in between seems to be Jhene Aiko. Keep in mind that repetition score only measures intra-song variation, so it should not be used to compare scores between different artists. Rather, it can show how often words are repeated within a song over the artist's entire catalogue of music.

4.2.4 TF-IDF Formulas and Calculations

In my calculations for Repetition Score and TF-IDF analysis, I look for unique word stems instead of unique words because we are more interested in the uniqueness of the base word than differences between verb tense of plurality of words. Word stems have the property of being words or word parts that are already converted to their common base form. For example, the words "help", "helped", "helping", and "helps" would all be converted to the stem "help". Not only does this reduce the size of the word list, it also considers different forms of words with the same meaning the same. I use the terms word and stem interchangeably in the below definitions since this analysis can be done on either the raw word list or the processed word stems.

Frequency of a word stem in a song is calculated by counting up the times that stem appears in the song's lyrics. Now that we have the frequency of each word or stem in the song lyric, you can calculate the Term Frequency (TF) for each term using Formula 4.2 below. This variable is different from the repetition score defined in the previous section because it is calculated on a per word basis instead of over a whole song's text. This means that each word in a song will have its own Term Frequency score.

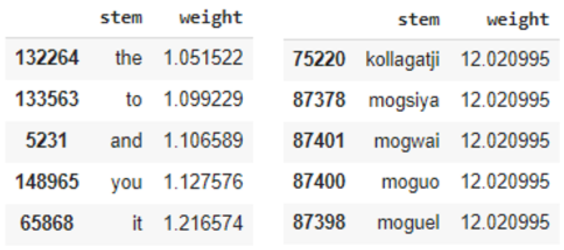
$$TF = \frac{\text{(Frequency of the word in the song)}}{\text{(Total number of words in the song)}} \tag{4.2}$$

Similarly, you can calculate the Inverse Document Frequency (IDF) as follows in Formula 4.3. Since the numerator is fixed for the number songs in our dataset, the IDF will remain the same for every word we are solving for. Essentially, the IDF equation is a statistic for a single word calculated over the entire collection of songs. Words that appear only in a few select documents (i.e. song lyrics) will have a higher IDF value since the ratio of total number of songs to songs with a rare word is higher. Taking the $\log_{10}$ of the ratio allows us to scale and compare the IDF scores for ratios that are very high or very low. These ratios refer to words that appear very rarely or very often, respectively.

$$IDF = \log_{10} \frac{\text{(Total number of songs in dataset} + 1)}{\text{(The number of songs containing that word} + 1)} + 1 \tag{4.3}$$

Note that adding 1 to the log term in Equation 4.3 ensures that there is no multiplication with 0 in Equation 4.4 and so the score is not 0. This way, terms that appear in all songs in the corpus will not be entirely ignored. Furthermore, adding the constant 1 to the numerator and denominator of the ratio in the log term ensures there are no divisions by 0.

See Figure 4.7 for a side by side comparison of the most repeated and the rarest words as computed by their IDF score. Each table lists the five most commonly appearing words and the least commonly appearing, respectively. On the left hand side table, there are no big surprises on what words appear. The words ”the”, ”to”, ”and”, ”you”, and ”it” are all extensively used in the English language. On The right hand side, the words are much more unique. The first two appear to be romanizations of Korean words. Songs with Korean words might appear in our dataset even though I tried to filter out only English Songs because many Korean pop songs have sections of their lyrics written in English. Therefore, our language detection filter used to filter out non-English songs in the Data Cleanup section may have considered it a song written in English. The last 3 words on the rare words list are proper nouns, so it makes sense that these obscure references wouldn’t show up in many of the songs in our dataset.



	stem	weight
132264	the	1.051522
133563	to	1.099229
5231	and	1.106589
148965	you	1.127576
65868	it	1.216574

	stem	weight
75220	kollagatji	12.020995
87378	mogsiya	12.020995
87401	mogwai	12.020995
87400	moguo	12.020995
87398	moguel	12.020995

Figure 4.7: Five Most Common Words (Left) vs. Least Common Words (Right) Used

The TF-IDF value is now just TF multiplied by IDF variable. Using Formulas 4.2 and 4.3 above, we can calculate the TF-IDF value using Formula 4.4. This is a useful metric because it is normalized by how often that word is used in the dataset or the entire collection of song lyrics.

$$TF\text{-}IDF = TF * IDF \tag{4.4}$$

Using this method, an TF-IDF vector for each song can be calculated by assigning a TF-IDF score for each word in the song. The scores are then assembled into an ordered n-dimensional vector, with n being the number words in the lyrics dataset. The TF-IDF vector for every song would contain n score values. Using our definitions, a value of 0 would represent the score for words that don’t appear in that song. Keeping the same vector ordering, we could now calculate TF-IDF vectors for all the songs in our dataset and compare them using mathematical methods. [6]

Implementing the TF-IDF vectors for each song in our database, we can find the word with the highest score in the n-dimensional dictionary of words. This represents the most unique words. In order to condense the TF-IDF vector into a single score for every song, we sum up all the elements to create the TF-IDF song score metric. In contrast to the Repetition Score that was mentioned earlier in Equation 4.1, this metric can be used to compare word variation between artists and between songs. See Figure 4.8 for a sample of TF-IDF scores from the same 10 artists in our dataset we referenced earlier. In this case, we see Jai Wolf again has the lowest average TF-IDF scores from the group. This means that not only is each individual song very repetitive in itself, the word choices in his songs are also very typical compared to other artists. Somewhat surprisingly, the album made by WWE wrestler John Cena and his cousin 'Tha Trademarc' had the highest average TF-IDF score, meaning their songs had more unique words than any other artist's songs in this sample.

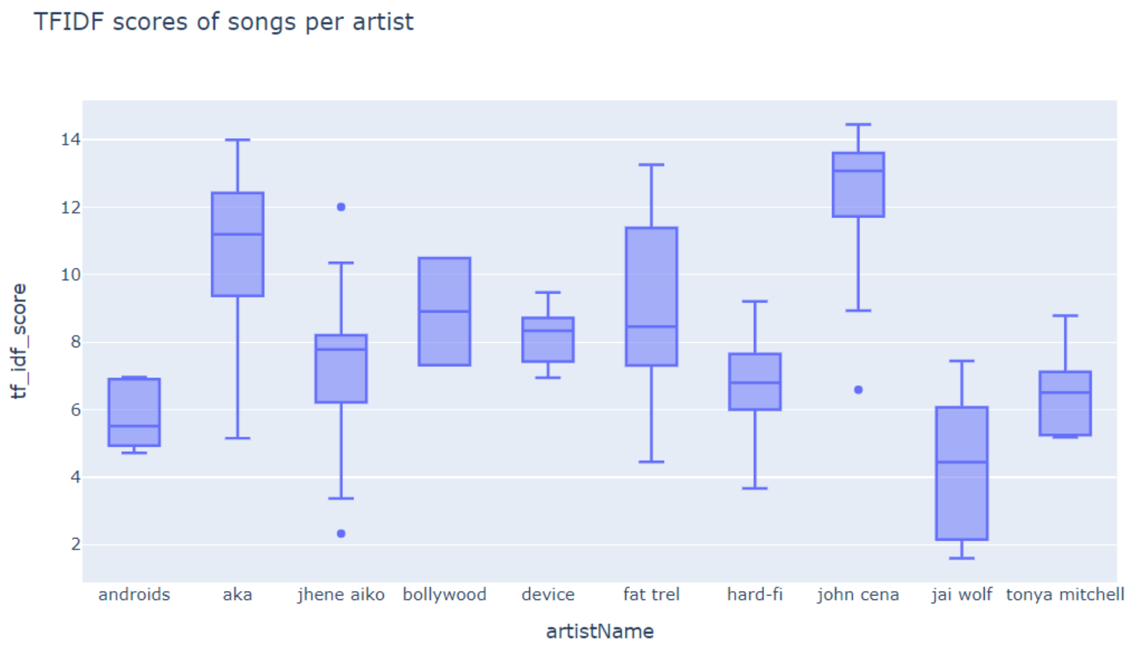


Figure 4.8: TF-IDF Scores of Songs per Artist

4.3 Generating Song Lyrics

First I tried to train on only country songs in my dataset to generate song lyrics. Since the we only have approximately 7200 songs in with this musicType in our song collection, the songs that were generated were very short or very repetitive. It did not fit mimic the creative structure of country songs that I was familiar with. The next thing I tried was to create songs out of the Rock/Pop musicType, since that was the most popular genre, with more than 33000 rows of song lyrics to train on. This generated better results, but i still felt that it had room for improvement. What ended up working best was a combination of several similar musicTypes into an umbrella category of songs. This umbrella category was composed of the Rock/Pop, Country, Manufactured Pop, and Folk musicTypes. I picked these styles of music because I wanted to mimic the Country Pop style of song lyrics. Using this method, I was able to collect around 43,300 songs to create my training and testing data.

I randomly sampled 90% of the Country Pop song corpus into my training dataset and left 10% in the testing or validation data. I used the 38900 songs in our test data to train the GPT2 algorithm on.

After a couple hours of training and generation time, the machine learning model created 200 samples of song lyrics for us. Some of the sample lyrics it generated were pretty convincing and read like real song lyrics without copying entire sections from any of the published songs. It can be hard to tell what is human written compared to machine generated at a glance. Take a look at the song lyrics on the next page. Did it come from a machine learning model or a human brain?

Always You

you can love the ocean and leave the sea,

you can hug the ghost of the ocean and spin it round and round,

you can fill the waters with a sense of magic,

and then you'll have to hold on to the scales of that world,

the magic that's inside, the scales of that world turn,

my sister would share some after, after you've tried the gallows game,

you can put me in a trance and make the wire

to sound and i'd never see her again, and she'd be the next one to fall,

you can tear this glass like you put your arms around me,

i had the taste in my cheek, and i knew you could drive me insane,

you could kill me and you just don't know it,

i would contribute to a pitiful effort and you will never,

you can waste my innocence and there will always be you,

there will always be you, there will always be you,

there will always be you, there will always be you,

there will always be you

Anonymous

If you guessed this was one of of the song lyrics that was written by the computer, you are right. In fact, a quick google search of the lyrics does not return matches for any other songs with the exact same lyrics. The text on the previous page has been formatted and reprinted for viewing. None of the words were edited. After reading this creation by the GPT-2 model, I named the song since none of the generated songs have names attached. Refer to the Appendix in Chapter 6 for more examples of interesting song lyrics that were completely machine generated.

4.3.1 Comparing Generated Lyrics to Test Data

I was able to generate 600 samples of song lyrics and had 4326 songs reserved in the testing dataset to compare it to.

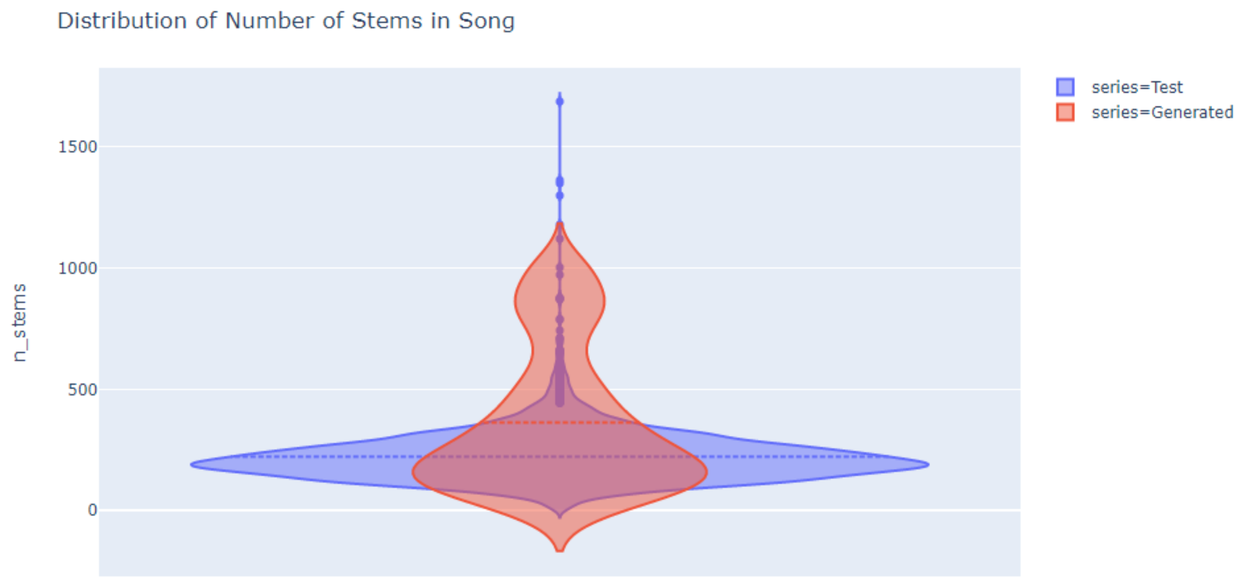


Figure 4.9: Song Length of Test Lyrics vs. Generated Lyrics

Figure 4.9 shows us the distribution of song length of the reserved test lyrics compared to the generated lyrics. As we can see, although there is some overlap in the distribution, they do not look very similar. The mean number of stems in the population of generated lyrics is 362. This is much higher than the mean of the validation song lyrics, which is 222 stems.

Furthermore, generated lyrics seem to vary much more in terms of number of stems and have a higher stem count on average than our human lyrics do. The test lyrics length is mostly a right skewed distribution with a median of 207 stems. This suggests that humans are used to producing something quick and concise while computer generated lyrics can suffer from not knowing when to stop when not given any top level constraints.

CHAPTER 5

Conclusion

The goal of this paper was to examine the process of generating novel song lyrics from the GPT-2 model software package using a set of previous lyrics to train on. We then analyzed the machine generated song lyrics and compared it to actual song lyrics.

I began with reviewing the history of using algorithms to create music and presented an overview of the transformer architecture used in our present machine learning model. In the Methodology chapter, we explained how the data was obtained and what variables were included in the processed dataset.

The final dataset included 122288 rows of curated song lyrics along with some cleaned metadata. Although this seems like a lot of song lyrics to train on, in practice this was sometimes not enough for training the GPT-2 algorithm. This was complicated by the fact that my attempts to generate lyrics through training only on specific genres or musicTypes further narrowed down the amount of lyrics available. The largest musicType category, 'Rock/Pop', accounted for only around 27% of our data.

In the Text Analysis section, we processed the song lyrics using word analysis models. The two metrics introduced were repetition score and TF-IDF score in order to assess the repetition in song lyrics and compare word usage between artists.

In the song lyrics generation section, I observed that the greater the amount of training data, the better the generated lyrics looked in terms of coherency and variation in sentences. The generated lyrics seemed to be very repetitive if the training sample went below approximately 15,000 songs. There are only 11,241 Urban/R&B songs and 7,216 Country sounds in our dataset. Although we had too few songs of the 'Country' or 'Urban/R&B' musicType to train on, we were able to work around that constraint. I resorted to combining several musicTypes with similar musical styles in the training data. This seemed to perform the best in terms of generating lyrics that had more contrast in words and sentences.

Then, we were able to compare the distribution of lyrics length from a validation set of lyrics to that of our generated lyrics. We found that computer generated lyrics generally had many more words than human created lyrics.

5.1 Further Works

Further work could focus on extending the corpus of song lyrics so that more than one musical genre would have enough training data to generate song lyrics. Furthermore, we can find a better metric for comparing the similarities between word vectors of the machine generated lyrics to that of the human created lyrics. Finally, there is a potential for using sentiment analysis on the song lyrics to characterize song lyrics from different musicTypes as well as the generated lyrics.

CHAPTER 6

Appendix - Examples Of Generated Lyrics

Best of You

$< |startoftext| >$ on a sunny morning when the lights go down,

the rest becomes your friend,

when you smile and no one does, the course's up and you have

no shame, no one,

but i hope we call it falling, i never realized that you cared,

there's no time to learn, surely you'll show up again soon,

but there's nothing left you know,

that's how it is, now baby if you ever change your mind,

please, tell me tomorrow, ooh, it's a beautiful day,

i just never realized that you cared,

you see the grass is fine, the stars are brighter but,

you always let the things that you do get the best of you,

wouldn't let them get me wrong, oh what,

i can't speak with you anymore, i never realized that you cared,

you see the grass is fine, the stars are brighter but,

you always let the things that you do get the best of you,

wouldn't let them get me wrong, well, again,

the grass is fine, the stars are brighter but,

you always let the things that you do get the best of you, wouldn't let them get me wrong

machine generated lyrics

0 Delusions

$< |startoftext| >$be a unified emperor in the skies,

come every morning and you'll see him there,

a chancellor of science, you'll be dumbfounded,

could come in the sun, without any doubt,

and usiga will ascend, a god with his birth, a conductor of time, in the mouth of a cryogenic cockroach,

some brilliance and neugety, would come, been done, undone,

0 is gone, this is never-ending, - within the mouth of the wannabe-a-no-one else,

a russi son, a leapsid smile, of the parody world, was the wannabe-a-nune,

the secret mouth of a faceless brick-bat, the words of every gullible person, directed upward,

levels of the electrophysi and meyco all spheres, in the groin of a cormorantard

initiates the dissolution of those tremors of the bili children,

the meaning of this contradiction. - i even learned that there are two religions,

that everyone knows the value of, and there is even a god,

there is no objective value, this is never-ending, - within the mouth of the wannabe-a-nune,

the words of every gullible person, directed upward,

levels of the electrophysi and meyco all spheres, in the groin of a cormorard

it initiates the dissolution of those tremors of the bili children,

the meaning of this contradiction. - unless can we find a breakthrough,

i could do anything, and here no one would hear me say,

there would be no one else to aid me, let's carry on to the end, for 0

Failed Dreams

$< |startoftext| >$ i think the standards of the city are really catching up,

i think that our adult society's getting to be really older,

when you step up, the kids move to suit, oh my,

nothing else matters, for they're in the front of the line,

and we are some of the least,

i know that your family's poor, and that's how you look at me,

i got a house worth watching, but we've got the tech to prove,

we need a dose of reality, we need the best city in the country,

i know that our adult society's getting to be really old, when you step up, the kids move to suit,

i think i can detect these changes in the institutions, the criteria of the city,

i think we've started a whole bunch of people going their own way,

we've thrown away some high-minded ideals,

i think we've gone and done some shit, we're in the middle of that shit,

i think we've dropped some cigarettes, and i get to a certain point where i believe,

that our fucking enterprise is dying,

the old corrupt office is crumbling, the old penthouse is spazzin',

burning up the sky and gliding you're dust keep you up all night,

no gonna get you'll get us,

i believe that our fucking enterprise is dying,

i think that our damn enterprise's dying,

i think that our goddamn enterprise is dying

machine generated lyrics

REFERENCES

[1] Marcus Du Sautoy. *The creativity code: art and innovation in the age of AI.* The Belknap Press of Harvard University Press, 2019.

[2] Jay Alammar. The illustrated transformer. *https://jalammar.github.io/illustrated-transformer*, 2018.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[4] Jay Alammar. The illustrated gpt-2 (visualizing transformer language models). *http://jalammar.github.io/illustrated-gpt2/*, 2019.

[5] Wikipedia contributors. Musicbrainz identifier: Using mbids for disambiguation. *https://musicbrainz.org/doc/MusicBrainz_Identifier*, 2019.

[6] Usman Malik. Python for nlp: Creating tf-idf model from scratch, Jul 2019.