Afraz Padamsee
Software Design
Professor Ben Hill
25 October 2019

MP3 Text Mining Writeup

Project Overview

The purpose of my project was to see if popular EDM songs share similar lyrical topics that are relatively generic and widely-applicable, with key-words onto which the listeners can project themselves while focusing on the track's instrumental. I mined the lyrics of 215 popular EDM songs using the API of the popular song lyrics website Genius.com to create the dataset. I analyzed the data by way of word frequency analysis (excluding meaningless words such as coordinating conjunctions) and n-grams analysis for sets of 2, 3, 4, and 5 words.

Implementation

I decoupled my code into 3 separate files that accomplish the 3 primary phases of the project: pulling the data, processing the data, and analyzing the data. The first file, called "textmining.py," contains everything that involves interacting with the Genius API. I pulled lyric data on a song-by-song basis, as the API was built around searching for songs individually, so this first phase of the project was to generate the function that would pull one song from Genius, which would then be repeated and expanded to create the whole dataset. The second file, called "songs.py," simply existed for the purpose of taking the functionality of pulling one song at a time and applying it to the list of all the songs whose lyrics I wanted to be a part of the analysis. I was originally just planning on choosing the songs by scraping the list of song names from Billboard EDM charts, but since they require a paid subscription to access the chart, I found other sources of lists for popular EDM songs and artists, and for those artists, I pulled the top 4-5 of their songs on Spotify that were not already on the other song lists. I used this list of 215 popular EDM songs and the lyric scraper function from "textmining.py" to pull all the lyrics. I then prepared the data for analysis by pre-processing them (removing punctuation, whitespace, making everything lowercase, etc.) and splitting them up into a giant list of words, which I wrote to a text file to locally cache it. The third file, called "lyricsmining.py," was where I ran the analysis on the full dataset of words created with "songs.py." I did another round of processing here, but it was mainly to narrow what was being analyzed, rather than to make the data analyzable, which was done in "songs.py." I then analyzed the new subset of data by word-frequency and n-gram frequency.

One area where I chose between multiple alternatives was how I chose to cache the scraped data. My two primary options were pickling the list of lyrics or writing them all to a text file. I went with the text file, as it gave me the option of manually looking through the stored data (in a text viewer) in addition to viewing it through Python, to find things I potentially might have missed when cleaning or processing the data. Pickling would have just served the purpose of locally storing the data, but writing it to a text file provided the added functionality of looking at the data in a different visual way. This actually proved to be useful, as I did look at the text file a couple times to quickly skim through the dataset when I could not figure out why certain words were or were not appearing in the analysis, which helped me clean the data and saved me a lot of time. For example, I would not have realized the dataset had a few Korean characters in it (as there were not many of them and I did not display the entire set in Python when I printed it) without looking at the text file. Noticing these characters gave me the idea to pre-process the data for non-english characters (e.g. letters with accents), saving me a lot of time that I would have spent if I caught it later on in the project and had to re-run all my analysis.

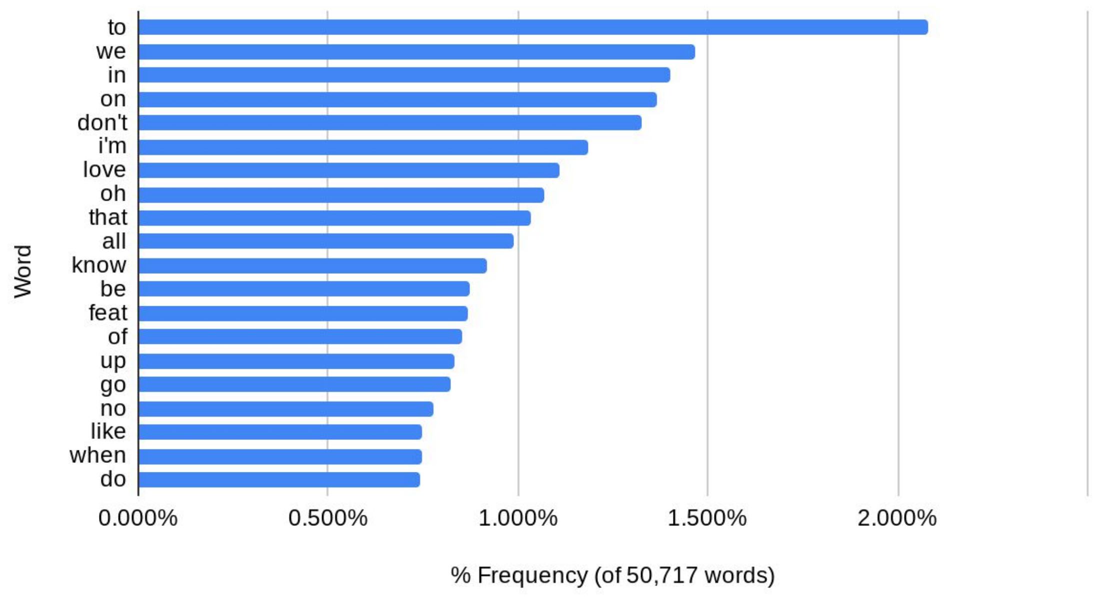
Results

Even though I tried removing “meaningless words” from the dataset before analyzing it, it appeared that most of the most frequent individual words (examined without their use in context) are ‘meaningless’. As visible in the word frequency graph, the top 6 words (before “love”) all hold no specific meaning purely by themselves, and this is after I omitted words such as coordinating conjunctions, articles such as “the”, “a”, and “an”, and most pronouns. I could have (and was seriously considering) adding even more words to the omit list while in the search for “unique words”, but I felt like I would be omitting too much, essentially removing all the actual highest frequency values in order to see what I wanted to see. 60% of the words in the top 20 most frequent may not have much meaning standing alone, but they have significant bearing on the context of the phrase or sentence in which they appear, of which the graph below shows nothing. So then, the choice of what to omit and what to keep becomes entirely subjective. I wanted the data to show me some of the longer words that are nouns or verbs, so I omitted almost all pronouns, even though those too change the meaning of the phrase in which they appear. “I” and “you” were the most frequent pronouns and the most frequent words overall. By omitting data to see what I wanted, I was missing an interesting finding staring me in the face.

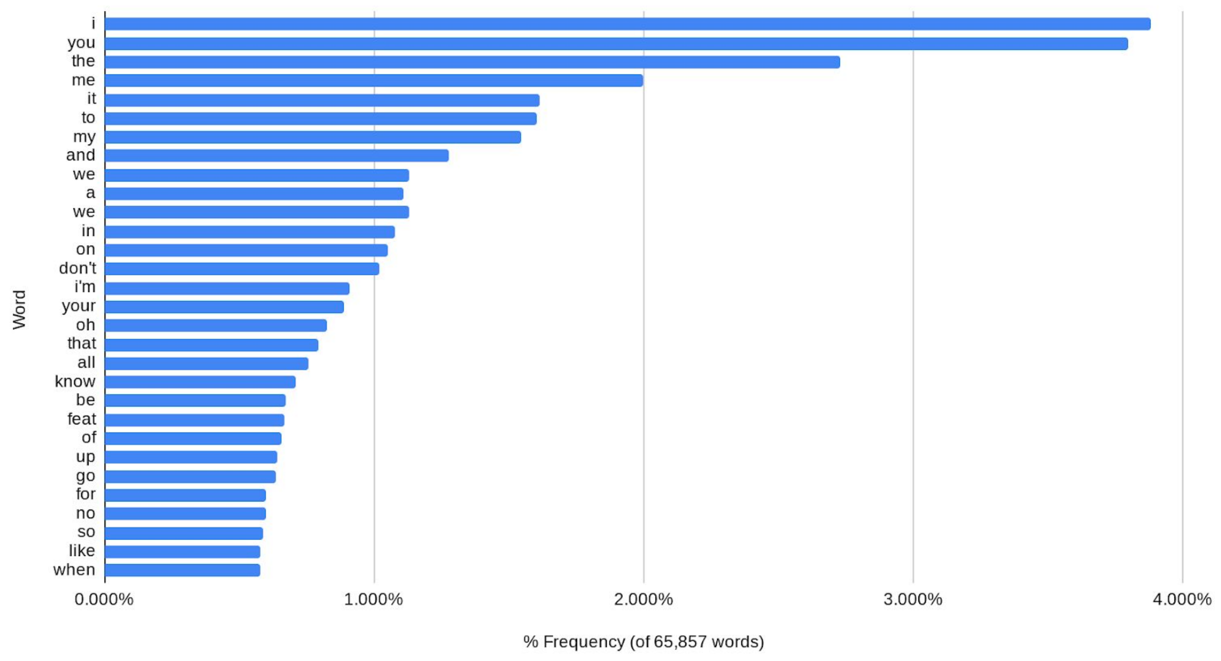
One of the truly interesting things about the output is how uninteresting it is. My initial goal to find out if EDM lyrics were generic came with my expectation of finding high frequencies of easily-self-projectible nouns (‘empty vessel’ words, so to speak) such as “love” and “feeling”; however, genericity of the lyrics had nothing to do with the potential genericity of the song topics or mood, but with the repetition of words so empty that they need a sentence to mean something, and can therefore be used in many ways.

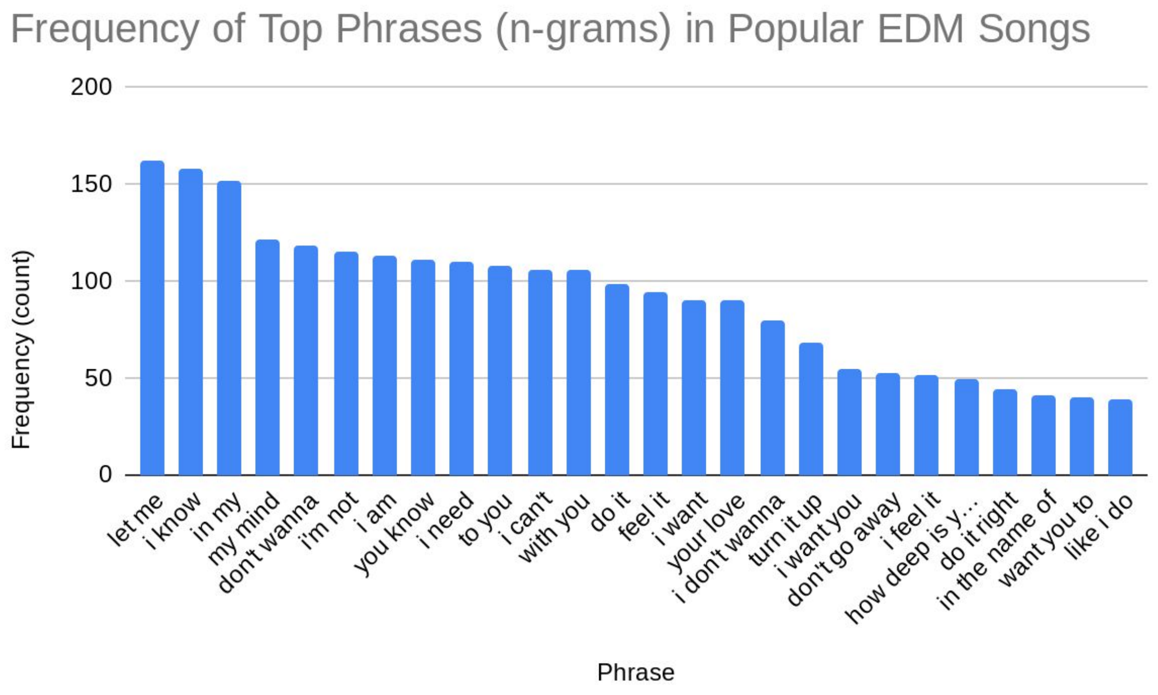
Another interesting thing that came from the n-grams findings was the prevalence of first-person pronouns such as “I” and “my” in the most frequent phrases. 8 of the top 12 phrases, or 67% of the phrases with frequencies of over 100, have a first-person pronoun in it, which is interesting as it pertains to the self-projectability of the wordage of EDM songs. When people sing the words at festivals, they say “I” and have an easier time projecting themselves onto the music so it resonates with them more. The psychology of language might be what is on display here, which underlies the genericity of the genre, which makes sense since basically the entire genre of popular EDM all has the exact same motive in mind: to make the listener “feel good”.

% Frequencies of top 20 Words in Popular EDM Songs



% Frequencies of top 30 Words in Popular EDM Songs (Unclean)





Alignment

For the most part, my initial goal aligned quite closely with the data and tools I used to carry out the project. My original intention was to prove that the subject (and therefore lyrical content) of most EDM is very generic, and I feel that I at least somewhat accomplished that. Upon working through the project, I narrowed my scope a little to look for generic and easily-self-projectable words and short phrases within EDM lyrics that theoretically would be indicative of generic song topics overall. After I decided I wanted to do something involving lyric data, I thought to myself: what genre should I look at? Should I look at hip-hop/rap? Hasn't rap lyric analysis been done to death? Is there anything new I can find? What other genres do I listen to a lot? EDM? What can I do with EDM lyrics?" And that's how I came to my idea! Before starting any analysis, the words I expected to see the most of were "love", "dance", "hands", "up", "girl", "boy", "drop", "feel/feeling", and "heart", and I was not expecting to see many swears at all. For phrases, I expected to see a lot of "Love you", "got me", "hands up", "all night", and "first time".

The data source itself has virtually no limitations in terms of what I was doing for this project, as it was all about lyrics and the source was just a hub of lyrics. The only glaring limitation with the data is the size and scope of the dataset itself. I tried to pick as representative a dataset as possible with the songs I picked, but the findings would definitely be more accurate and paint a better picture of the EDM genre as a whole if hundreds or even thousands of songs' lyrics were included in the dataset. Still though, I was able to draw some interesting insights I did not expect and find some pretty common loosely-emotional words, such as "love", which supported my "EDM is generic" pseudo-hypothesis.

For the word frequency analysis, dictionaries were an absolutely essential part of the process, as utilizing them to create a ‘histogram’ of word appearance frequencies was the fundamental process with which my analysis was possible. In terms of the answers the dictionary tool served me, I was able to use them to get the program to output exactly what I was looking for. My process of frequency analysis with n-grams however, was not so perfect. I still got good, accurate, and interesting results from the analysis, but there were some limitations that I had to accept. The tools I used for the analysis were the nltk.util and collections modules, to run the n-grams and count the frequencies, respectively. The primary issue that I had was that the n-grams function was grabbing multiple parts of the same phrase (e.g. for n=2 and the phrase being “let me love you”, n-grams() would count “let me”, “me love”, and “love you” separately every time that one phrase appeared, so when I displayed the top 10 most frequent 2-grams, 3 of them would be all from the same phrase with the same frequency), which just makes the output more difficult from which to draw conclusions. This difficulty is exacerbated by the fact that a lot of the most frequent 2 and 3-grams were small parts of bigger phrases that also appeared often enough to be in the relevant part of the output, bringing about the issues of double-counting phrases by keeping them both in, or potentially skewing data by removing one of the phrases. Even though multi-counting the same phrase and different parts of a phrase are limitations of this analysis tool, they did not derail my project. I was still able to efficiently analyze the data and have it output almost exactly what I wanted.

Overall, I am pretty confident in the answers my analysis gave me. I feel that in terms of what my dataset was, the analysis tools were almost entirely accurate, but I cannot say precisely how indicative the findings for this dataset are for the entirety of popular EDM.

Reflection

Some things that went well were the pulling of data from Genius and the use of dictionaries to find the word frequencies. I talked about both of those earlier so I will not get into great detail here, but they worked mostly without any problems, and they did for me almost exactly what I expected them to do from the onset of the project. There are some things I could have done differently or improved upon that might have made the code more efficient, such as using Filtering to process the data in “lyricsmining.py”, and using pickling instead of file writing to locally cache the data, so (especially for the former) if I were to do this project again, I would definitely consider implementing those disciplines into my code. On top of that, despite searching extensively for a fix, I maintain that there has to be a better way of generating the list of songs from which to pull lyrics. I did not realize going into the project that EDM might have been the single worst genre to pick in terms of cooperating with code to pull song titles from the internet: many EDM songs are collaborations between multiple artists, and the combination of all their names gets recognized as the ‘artist name’, rather than a list of multiple artists, which makes pulling songs by artist basically useless, since this is very common in EDM (and I was picking songs by artists since I could not access the Billboard charts). Also, EDM has lots of popular remixes of other EDM songs, so out of an artist’s top 10 songs, 4 of them might be lyrical duplicates of other songs in those top 10, again, making pulling by artist or even pulling of long song lists not very efficient, since I would have to manually delete the duplicates. Issues like these have rendered ineffective the song scraping programs I could find, but I still think there is something out there that works, so if I were to do this project again, I would take more initiative earlier in asking for help to find a program or method that works despite the features of the EDM genre. Doing this would in essence help mitigate the data-related limitations of the program, since it would allow me to generate a much bigger list of songs at a much faster rate, growing the dataset and making it more representative of the genre as a whole. A way I could mitigate the tool-related limitations of the n-grams function is by creating a function that looks up and counts a user-inputted phrase in the giant word list, rather than outputting the most frequent n-grams. This would bypass ngrams() entirely and all of the limitations associated with it. What's great about the program I built for this project, is that it can work for literally any set of music: a genre, a specific artist, an album, a time period, a specific songwriter, etc.. I would also be curious to have this exact program run on other genres to compare the relative frequencies of pronouns and 'meaningless words' to those in EDM. The program also has interesting applications but can be taken further in various different directions based on the programmer's goals, such as sentiment analysis and Markov synthesis.