BrainStation Capstone: Lyrics and Popularity

Justin Ng
April 11, 2023

THE PROBLEM

Song popularity can be influenced by many factors, but one factor has garnered less interest for consideration when predicting song popularity. This factor is the **lyrical content** of the song. This lack of interest could stem from the extra work required for processing text data and the fact that different genres can have differing vocabularies. Despite these difficulties, this project aimed to see if it was possible to **predict a song's popularity solely using the lyrics alone**. Limiting ourselves to only lyrics will make this objective harder to achieve, but being able to meet this objective would bring value to the songwriting industry. Specifically, our findings could enable individuals to **write popular music more efficiently**.

We will now share our journey in trying to unlock the relationship between lyrics and popularity.

THE DATA

The dataset used was originally from Cornell University where they scraped the lyrics of approximately **35000 songs from Genius.com** from 2019 to 2020. This dataset also provided release years, primary artist, song titles and **Genius.com page views, which we used as a proxy for popularity.**

THE CLEANING

Once we found the dataset, we faced our first challenge, which was to clean the lyrics. We decided to manually clean the text, as current packages were struggling with the format of the lyrics. We had to make custom functions to deal with tags within the lyrics and also removed all the punctuation.

Another issue that occurred during this cleaning process dealt with the **existence of non-english songs** in the dataset. We decided to use the fasttext package and a pretrained language model to try and detect all the english songs. This resulted in the detection of **26 languages** and a reduction of the dataset to approximately **33000 songs**. We also decided to remove stop words and stem the lyrics before moving on to the exploratory data analysis.

After the lyric cleaning we also needed to find ways to represent the lyrics as vectors, so we decided to start with three approaches: **Countvectorizer, TF-IDF and LexVec Embeddings.**

MOVING AGILE

When looking into our proxy for popularity, the Genius.com page views, we realized that the **values were highly skewed**, we decided to instead transform our target to get a less skewed distribution. With this transformation completed we decided to try some preliminary modeling. Unfortunately, our first linear regression models produced results that were worse than just predicting the mean page views. With this result we decided to pivot and **change this problem to a classification problem**, by separating the **popularity into thirds** and designating high, medium and low popularity. After this pivot our preliminary modeling demonstrated that we could not get prediction accuracies much higher than **10% over the**

baseline regardless of lyric representation we used. This made us rethink our dataset and our proxy for popularity.

MORE MODIFICATIONS TO THE DATASET

We decided to change our target to be more in line with a song's popularity, specifically we decided to use Spotify's API to scrape for the **Spotify popularity rating** and append this to our dataset. This Spotify popularity rating is a number from 0 to 100 that represents how popular a song is on the platform. We once again split the data into three groups. The correlation between Spotify popularity and Genius page views was low, so we hoped that **this new target would be a better proxy for a song's popularity.** During this scraping process we also realized that we did not take into account that the same song could be performed by different artists. We decided to take the first occurrence of a song in the dataset, regardless of the artist, for simplicity sake.

We also added the genre of a song to the dataset to see if lyrics were more **predictive of popularity for different genres**. This scrape, and adjustment for unique titles, reduced our dataset to around **28000 songs**. We also added an additional lyric representation using the OpenAI API. We transformed each lyric with a second generation OpenAI text embedding model known as Ada.

MODEL TUNING

From our original text representations we decided to move forward with only TF-IDF for our text representations. TF-IDF was selected as there was no definitive performance benefit to using the other representations. Additionally, we believe TF-IDF could capture more of the unique words in song lyrics better than a Countvectorizer. We also wanted to compare the results of TF-IDF with the Ada embeddings. **Table 1.** outlines our results from tuning our various models.

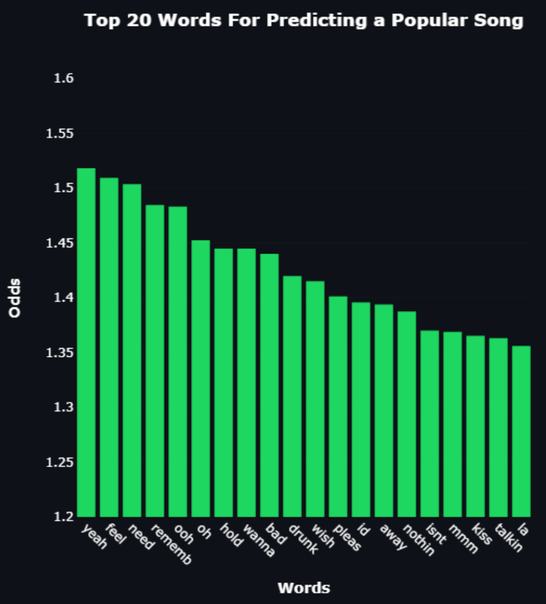
Table 1. Modeling Results After Tuning

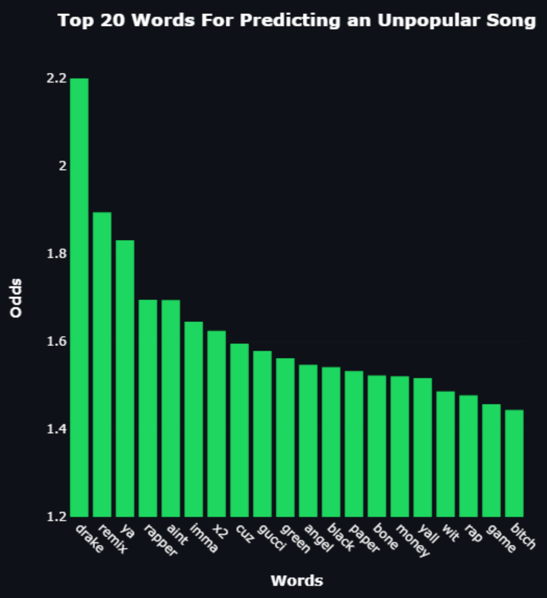
Model	Text Transformation	Test Accuracy	AUC of Micro-Average ROC Curve
Logistic Regression	TF-IDF	0.42	0.60
	TF-IDF + NMF	0.40	0.58
	Ada Embeddings	0.45	0.63
	Ada Embeddings + PCA	0.42	0.60
	TF-IDF + Hip Hop Only	0.43	0.62
Multinomial Naive Bayes	TF-IDF	0.42	0.60
	TF-IDF + Hip Hop Only	0.45	0.63
Random Forest	TF-IDF	0.42	0.60

From the above we can see that we have similar weak classifiers across the board, regardless of the model and text representation. There was a marginal performance boost when we subset the data and only look at hip hop songs. Similarly, the Ada embeddings also provided a small increase in performance. Surprisingly even this more complex text transformation method did not perform better than a simpler method such as TF-IDF, which demonstrates the difficulty this problem poses.

MODEL EVALUATION AND THE CURSE OF “DRAKE”

Even though we had created some weak classifiers we were still interested to see what words were most important for predicting a certain popularity class. Interpretability was difficult with the Ada embeddings so we decided to try interpreting the **logistic regression model that used the TF-IDF transformed lyrics as input**.





We found that words that increase the odds of the model predicting a popular song are mostly positive and deal with physical contact (**kiss, feel and hold**) and partying (**drunk and LA**). In contrast, words that increase the odds of the model predicting an unpopular song (low popularity class) deal with **themes prevalent in music created by novice “SoundCloud rappers”**. Surprisingly, mentioning **Drake** in a song seems to increase the odds of an unpopular song by 2.2 times. This large increase could once again be fostered by novice rappers using Drake’s name in their songs. We also **implemented this weak classifier in a Streamlit app** where individuals can try their luck at making the next hit. The app also provides avenues to explore the dataset in more depth.

FUTURE DIRECTION

Through this journey we have discovered that it is difficult to predict a song's popularity with lyrics alone, but there is still some predictive power.

Future steps could involve being more granular with lyric analysis. Specifically, this analysis only looked at the word frequency within the entire lyric. In contrast, we could improve this analysis by including specifics at the chorus, verse and hook level. Additionally, we found that lyrics were more effective in predicting popularity when we controlled for the genre. This supports the idea of controlling for more features to eliminate the noise in lyrics from vastly different genres of music. These improvements to our analysis will hopefully allow us to gain a deeper understanding of lyrical content in the songs of the future.