ORIE 4741 Midterm Report

Aliyah Geer, Adarsh Mital, Elise Kronbichler

October 2020

1 Introduction

Since writing our initial project proposal, our group has shifted its project objective from a more politically-centered analysis to a purely emotion-centered one. **Our new question is**: Can we successfully predict the mood of a song based on the lyrics?

To better fit accommodate this objective, we changed our main dataset from the "million song dataset" to the "Moody Lyrics" dataset, which features about 2000 songs that have already been labelled by four main sentiments: "happy", "angry", "sad", and "relaxed". The features of this dataset were not extensive, only the title of the song and the labelled mood. To get the lyrics to each of these songs, we used a package called "lyricsgenius", which scrapes lyrics of a specified song from the popular site, Genius.com.

2 Preprocessing

Before anything else, we needed to clean the lyrics dataset. Here, we had to address a multitude of issues: null values, upper-case words, non-english lyrics, "stopwords", and punctuation marks.

We created a preprocessing function to address these issues which converted all letters to lower-case, eliminated all rows with null values and non-english lyrics, as well as any punctuation marks and "stopwords" (words common to all songs e.g "I", "and", "that", "the"). We found a package to handle the latter issue. After doing these things, we had a cleaned and comprehensive set of lyrics to work with, with over 20,000 words and about 1900 songs.

3 Bag of Words

For our preliminary analysis, we decided to use a bag of words model rather than TF-IDF, which takes into account relative frequencies and weighted them. We thought that the BOW approach was the simplest to first get a general idea of our data, and then continue from there. To start our model, we used sklearn's

CountVectorizer function, which counts the frequency of words and selects the most frequent words as features. We were able to choose how many words we wanted to use as features via an argument in this function.

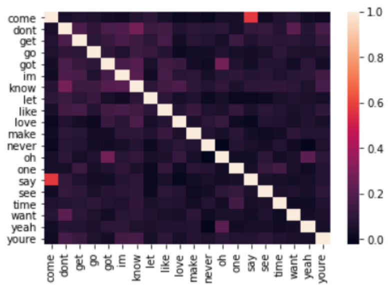
4 Naive Bayes

To get a "base" accuracy, we used a probability-based text classification function called Gaussian Naive Bayes. This function looks at the probability of each class, or in this case each mood, as well as the probability of the features (words) given the class. Naive Bayes then uses that to find the probability of the class given the features. This function performs better than other models such as logistic regression because it requires less training data, and considering our 2000 song data set, we did not have much for training data. We used the sklearn function train_test_split to allocate our training and testing data sets.

To determine the optimal number of features to use with Naive Bayes, we used k-fold cross validation to get an accurate estimate of the prediction accuracy of each number of features in a set of predetermined feature numbers (from 1000 to 10000). We found that around 7000 features allowed us to get the highest accuracy on our testing data, which was about .534. On our training data, we got an accuracy of about .90, which implies that our data was overfitted. We would like to amend this with our next model, as outlined below.

5 Correlation Matrix

Before proceeding with fitting linear models that assume linear independence among variables, we wanted to look at the relationship between our features(word counts). Here is a correlation matrix between the 20 most common words in the data set.



This is an encouraging sign since these words seem to be somewhat independent from each other. However we have not looked at correlation amongst our less common words, which could reduce the precision of our estimated coefficients for a regression model we fit in the future. We fitted a non regularized logistic regression model with k fold cross validation and received a test set accuracy of around .45 which indicates there is room for improvement here.

6 What's Next?

As we continue with this project, we would like to move away from BOW and instead focus our efforts into using a neural network to get embeddings of our lyrics set. This embedding would incorporate context within the lyrics that is not found in a BOW model, in terms of the ordering of the words. This model should (hopefully) prevent overfitting, and give us theoretically the best accuracy possible. We are looking at specifically using BERT, a pre-trained NN developed by Google. We also would like to try using TF-IDF (better than BOW), and compare our results to BERT. Finally, we will investigate which loss function is best suited for our dataset, and try to balance bias + variance through regularization to reduce overfitting, and go from there.

7 Visualizations

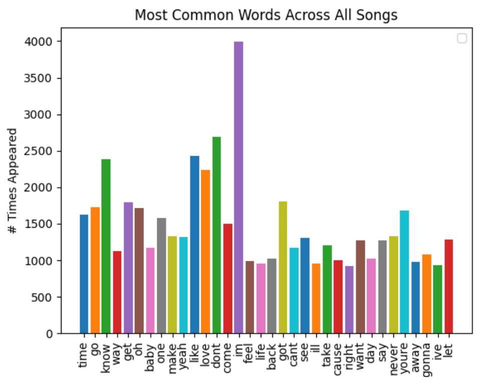


Figure 1: most commonly used words

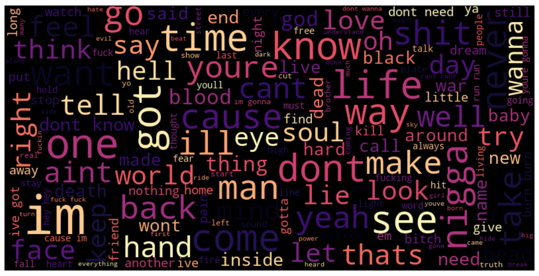


Figure 2: wordcloud for songs labelled "angry"