CIS 545 Final Project: Understanding the Influence of Music

Jeffrey Cheng, Eric Dong, Arnab Sarker

May 2, 2018

Contents

I. Introduction

I. Problem Domain

The influence of music is eminent in our society. The music industry currently makes over $130 billion USD per year in recorded sales, but more can be done to have a data driven understanding of what can make a song popular. Musical artists around the globe are constantly producing new songs that get played on the radio and streamed on online services, providing massive amounts of data ready for analysis.

Most songs can be broken down into two parts - a vocal line and an instrumental line. In this report, we begin with some exploratory data analysis to understand aspects of songs that could be related to popularity and influence, and then try to understand how both vocals and instrumentals impact the influence of a song.

When understanding the vocals of a song, we analyze the way in which lyrics impact the popularity of a song. In pop culture, many songs with repetitive lyrics tend to be played on the radio, such as Lil Pump's "Gucci Gang." On the other hand, songs with a variety of lyrics, such as Queen's "Bohemian Rhapsody" have also achieved levels of popularity, so it is difficult to understand the relationship between a song's lyrics and its eventual popularity.

Lyrics are not all that there is to song writing. Rappers such as Twista may arguably have technical prowess in their lyricism and songwriting abilities, but don't exactly have the cultural dominance that the Beatles did in their prime. Another major aspect of songs is the instrumentals. We thus try to understand the impact of instrumentals of music by looking at the way in which songs sample one another. In music, sampling occurs when one takes a piece of one song and then interpolates ("samples") it into another song. By analyzing the relationships between samples in a song, we can construct a directed graph, and then use pre-defined notions of graph centrality to understand what songs can be described as "influential" in terms of its impact on other songs, and why.

After the acquisition of large data sets, through Genius and Spotify, we run efficient algorithms to understand what makes music influential.

II. Problem Description

We divide our problem description into three main parts, with the overarching theme to be understanding what exactly can make a song influential.

1. Find features of songs that will help us understand their influence. In particular, we believe popularity, danceability, and energy will create clear genre clusters among new genres, and that older genres that have influenced multiple families of music may lie between clusters.

2. Determine which songs based on the graph of sampling have influence due to their instrumental tracks. We hypothesize that older songs from genres such as the blues and funk will be influential in the production of newer songs in hip hop and R&B.

3. Understand the relationship between song lyrics and the measures of popularity. Our hypothesis is that lyrics will be able to predict popularity to some extent, as songs that are "catchy" tend to be more popular.

II. Data Preprocessing

I. Datasets

We use two data sources at the individual song level: the Genius API[1] and the Spotify API[2]. From the Genius API, for each song, we scrape its name, artist, lyrics, and audio sampling references

[1] `https://docs.genius.com/`

[2] `https://beta.developer.spotify.com/documentation/web-api/`

using the external lyricsgenius[3] module. The audio sampling references field refers to other tracks that the given song samples from. To determine which songs to include in our dataset, we first select the top ten songs for each of the top twenty artists for each possible starting letter of the artist name, with non-alphabetical characters being considered as one group. For example, for all artists in the Genius database[4] whose names start with the letter "A", we choose the top twenty artists and use their top ten songs based on Genius's determination of popularity. We then populate the rest of our dataset by randomly selecting songs in Genius through their Genius API identifier while ensuring there are no repeated entries. The motivation behind this two stage scraping approach is so that our dataset is diverse and has a variety of both popular and unpopular songs. The final Genius dataset contains 12343 rows.

From the Spotify API, for each song, we scrape its popularity, danceability, and energy using the external spotipy[5] module. The latter three fields are Spotify audio feature metrics. According to Spotify, "popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are" while danceability and energy describe how danceable and intense a track is, respectively. The songs we use in our Spotify dataset are all tracks from our Genius dataset that also exist within Spotify, based on matching song titles and artist names. This leads to a Spotify dataset containing 5510 rows. The following table describes the information we pulled from each source.

Genius API	Spotify API
Song Name, Artist (**primary key**)	Song Name, Artist (**primary key**)
Lyrics	Popularity
Audio Sampling References	Danceability
	Energy

II. **Data Wrangling and Integration**

Given Genius track data with schema {Title, Artist, Lyrics, References} and Spotify track data with schema {Title, Artist, Popularity, Danceability, Energy}, we join on Title and Artist to get a dataframe with the desired structure.

Title	Artist	Lyrics	References	Popularity	Danceability	Energy
String	String	String	String	Int[0,100]	Int[0,1]	Int[0,1]

More specifically, our joining procedure first searches for a nearly approximate match in Title and Artist and removes songs that do not show up with a match in Spotify. A song within the Genius dataset matches a song within the Spotify dataset if the names are exact matches after removing capitalization and special characters and the artist based on the Genius data is one of the artists based on the Spotify data. This method allows us to catch edge cases such as "Flawless Remix" by Beyoncé and Nicki Minaj, which appears as "***Flawless (Remix)" by Beyoncé in Genius.

III. Genre Clustering

I. **Modeling**

We begin our analysis by performing *k*-means clustering on the popularity and audio feature portions of the Spotify data. The goal of this analysis is three-fold. First, clustering will help us better visualize and understand the distribution of our data. Second, we hope to define more meaningful genre clusters than how music is currently being classified into genres. There are many instances of songs that fall into very different genres and yet invoke similar emotions. For example,

[3] https://github.com/johnwmillr/LyricsGenius

[4] https://genius.com/artists

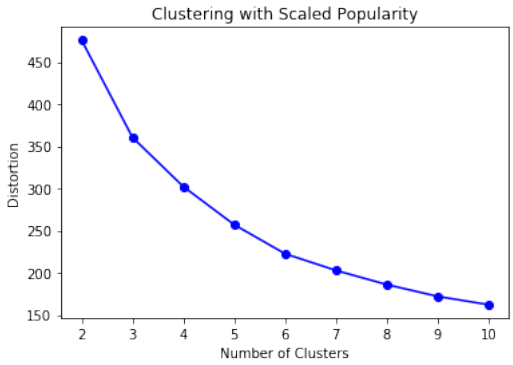
[5] https://github.com/plamere/spotipy/blob/master/docs/index.rst

Pachelbel's Canon by Johann Pachelbel (classical) and Levels by Avicii (EDM) both tend to invoke rising feelings of positivity and movement. Clustering on audio features and popularity may also be more robust since genres tend to shift over time. Lastly, before predicting the popularity of songs from their lyrics, we want to explore if danceability and/or energy are associated with a track's popularity. While lyrics and instrumentals can play an important role in a song's success, it makes sense that a song's audio features can also have an impact on its popularity.

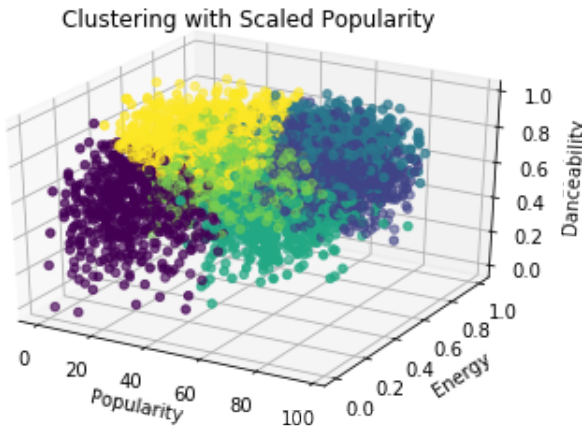
Because we are not using a predetermined number of genres, our value for k is unknown initially. Thus we find the optimal k by testing a range of k values and plotting their level of distortion to determine the elbow. Additionally, we rescale the popularity feature so that it lies within the same range of zero to one as danceability and energy do.

II. **Results**

After creating the clusters for each k, we plot the distortion under each k:



From this, we see that the elbow occurs roughly around six clusters. Proceeding with $k = 6$, our k-means clustering results in the following cluster visualization:



The table below summarizes the cluster centers and provides an example song in each cluster:

Cluster	Popularity	Energy	Danceability	Example Song
1	0.1603	0.3044	0.4670	Nocturne in B major by Frédéric Chopin
2	0.5869	0.8003	0.5560	Congratulations by Post Malone
3	0.6409	0.6148	0.7727	Havana by Camila Cabello
4	0.6012	0.4129	0.5104	Perfect by Ed Sheeran
5	0.1590	0.8683	0.4086	Radical Love by Victory Worship
6	0.2089	0.6616	0.6964	Django by Dadju

Our clustering analysis shows that our data contains songs that span the entire range in terms of popularity, energy, and danceability. Furthermore, it appears that energy and danceability are not particularly correlated with popularity.

Our clustering analysis identifies six distinct groups based on popularity and audio features. While these clusters are not necessarily useful, they are interesting because they illustrate how songs from different genres can still "feel" similar. The following table names these segments in our attempt to characterize the clusters:

Cluster	Description
1	Obscure and mellow
2	Mainstream party music
3	Popular dance tunes
4	Soft hits
5	Intense niche songs without a beat
6	Everyday flops

IV. Instrumental Influence Graph

I. **Modeling**

To understand the influence of a song based on its instrumental, we use notions of graph centrality. First, we form a graph based on which songs sample others. This data is given by the Genius data set, as there is a section of the JSON object that lists songs that are sampled by a particular song. For example, according to Genius, for Kendrick Lamar's "Blow My High (Members Only)," there are three samples, taken from Jay-Z's "Big Pimpin'," Aaliyah's "4 Page Letter," and Dexter Wansel's "Voyager." In our graph, we represent each song as a node, and then we have an arrow from song A to song B if song B samples song A. In the example above, the three samples would have outgoing edges to "Blow My High (Members Only)." This convention is used to maintain consistency in that "sources" are seen as sources of music as well as sources in the graph theoretic sense.

We then wished to determine an appropriate measure of centrality for our graph. For this analysis, we chose to use Kleinberg centrality[6] to understand in what way a song was influential. Kleinberg centrality provides two metrics for the centrality of a node: a measure of "hub centrality" and a measure of "authority centrality." A "hub" in a graph refers to a node that points to nodes with high authority - in the network context, this can be thought of as a "hub" of information, where a node with high hub centrality points to many nodes with important information. An "authority" in a graph refers to a node that is pointed at by eminent hubs. From these recursive definitions, we can use the structure of a directed graph to determine which songs are importantly used as samples (our "hubs") and what songs have influential samples (our "authorities").

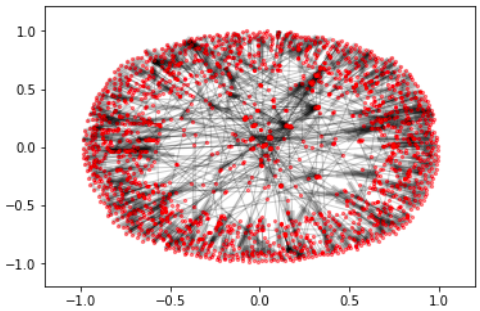
We chose to use Kleinberg centrality for multiple reasons, the main reason being its common use in citation networks. With our graph structure, it would be bizarre to see a cycle, as songs tend to sample songs that have come before them, and not songs that will be made in the future. Kleinberg centrality is designed for citation networks, which have this same directed and acyclic graphical structure. In addition, the updates in Kleinberg centrality are iterative, similar to the calculations for PageRank, allowing for a distributed approach for the calculations of hub and authority centrality. In each iteration, the hub centrality is measured as the sum of the authority centralities of the out-neighbors of a node, and the authority centrality is calculated as the sum of the hub centralities of the in-neighbors of a node. Normalization is done so the values can convege, and the code is easily implemented in an efficient way that takes advantage of optimized SQL queries.

[6] https://www.cs.cornell.edu/home/kleinber/auth.pdf

The data itself came primarily from the Genius data set, as this provided valuable information for what songs had samples. In addition, because an exploratory data analysis found only a thousand sample relationships in the original data set, a breadth first search was run in order to find more sampling relationships, taking samples of the songs that were already in our data set and then adding them to the data set, making our graph slightly larger and adding more layers of structure to the citation network.

II. **Results**

After the graph was created, we generated a rough visualization of the graph's structure:



From the graph above, we see that the graph of samples is relatively sparse: With over 1,800 nodes, there are only approximately 1,500 edges, whereas a complete graph would have over 2,000 times as many edges in it.

Once Kleinberg centrality was determined based on the iterative process described above, the authority values for the top 10 "authorities" of the graph were as follows:

Artist	Song Title	Authority Centrality
Eazy-E	Eazy Duz It	0.495
Doug E. Fresh & The Get Fresh Crew	La Di Da Di	0.0470
Eazy-E	Boyz-n-the Hood	0.0356
Doug E. Fresh & The Get Fresh Crew	The Show	0.0356
Sly and the Family Stone	Sing a Simple Song	0.0355
The Honey Drippers (70s Funk)	Impeach The President	0.0325
James Brown	Funky Drummer	0.0325
Ohio Players	Funky Worm	0.0309
LL Cool J	Mama Said Knock You Out	0.0300
James Brown	Funky President (People It's Bad)	0.0294

We have the following values for the top 10 "hubs" in the graph:

Artist	Song Title	Hub Centrality
Eazy-E	Eazy Duz It	0.709
Ice Cube	No Vaseline	0.0382
Redman	Time 4 Sum Aksion	0.0375
N.W.A.	100 Miles and Runnin'	0.0367
LL Cool J	Mama Said Knock You Out	0.0353
N.W.A.	8 Ball	0.0317
Beastie Boys	Time to Get Ill	0.0312
Kriss Kross	Jump	0.0293
N.W.A.	Gangsta Gangsta	0.0292
Bruno Mars	Grenade	0.0289

Note that in the hub centrality table, an entry for "8 Ball (Remix)" was omitted as it had the same information as the entry for "8 Ball." From the above tables, we have a few major conclusions.

First, we note that there exists a node with relatively large authority and hub centrality - the song "Eazy Duz It" by Eazy-E. Because this song is an outlier, we manually inspect it, and see that this song samples 15 individual songs, and then is sampled by 23 songs. Most songs in the data set that sample songs will only sample 1 or 2 songs, so it is unsurprising that "Eazy Duz It" commands so much of the centrality in this data set.

We also see that, as expected, the songs that have high authority centrality are generally older and from the genre of funk - two of the songs in the top 10 are by artist James Brown, who is known as a major funk artist. Also as expected, we see that the songs with high hub centrality are more modern, at least compared to the songs with high centrality. For example, "Grenade" by Bruno Mars is included among the songs with high hub centrality.

However, one unexpected result of the data analysis is that there are a couple songs that have both high hub and authority centrality, as well as songs from the same artist in both charts. This likely comes from the recursive nature built in to the definition of Kleinberg centrality. These nodes likely have high "betweenness," in which they have a high number of in neighbors and out neighbors in the graph. This is clearly the case for "Eazy Duz It," and suggests that artists such as Eazy-E and LL Cool J were not influential in just using samples of popular songs, but the songs made by these artists were influential in their own right.

Comparing these values to the metrics for popularity, we see that there is not a strong relationship between our centrality measures and popularity. Merging the Spotify popularity data with the centralities, we have the following plots:

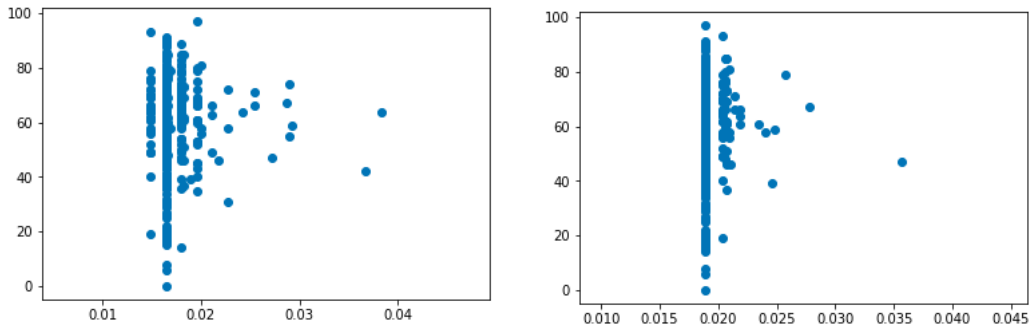


Figure 1: *Scatter plot of hub centrality against popularity (left) and scatter plot of authority centrality against popularity (right)*

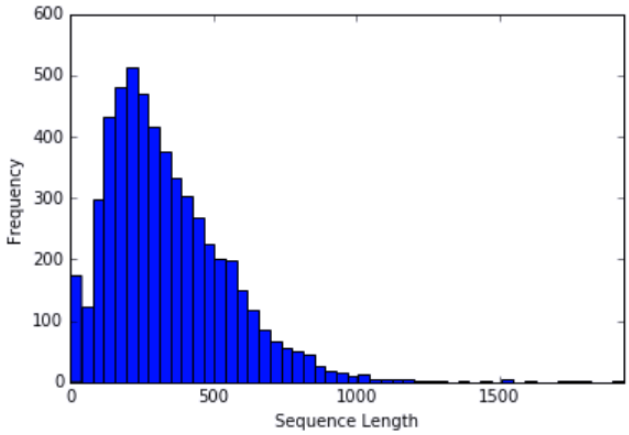
Hub centrality and popularity have a correlation of -0.009, and authority centrality and popularity have a correlation of 0.002, implying no relationship between either centrality measure and popularity. Ultimately, the above results were expected, as the centrality calculation does not take into account the popularity of certain songs. In fact, this lack of correlation gleams an interesting deduction: Songs that influence other artists don't necessarily influence the general public.

V. Popularity Prediction

I. Featurizing Lyrics

The Genius API gives each song's lyrics as a single string, which is difficult to model. We featurize the raw lyrics for use as a predictive parameter.

- Split the lyrics by whitespace and remove special characters to obtain a list of alphanumeric words.
- Featurize each word.
  - We use a word embedding model called "Word2Vec". The model is trained by running a window over text and predicting the center word from context. We utilize Stanford's GloVe[7] model, which contains 400K words, each of which is represented by a feature vector of length 50.
  - Apply the GloVe transformation to each word in each song. For unknown words (i.e. words not in GloVe), substitute with the unknown feature index (399999).
  - Pad / trim the song lyrics to the same length. We find that the song with maxmimum length is 1943 words; every attempted model on this length failed, likely due to the insufficient number of LSTM units to length. Given the distribution of song lengths:



We pad/trim the songs to a length of 200 songs.

Our final schema for lyrics $\Longrightarrow$ popularity modeling is then:

Title	Featurized-Lyrics	Popularity
String	Array_{200} of Int[0, 400K]	Int[0,1]

II. Modeling

We are trying to find a regressive model that takes variable-length feature vectors of words as input to predict a real number in the range $[0, 1]$. The temporal nature of variable length vectors and the complexity of word contexts is suggestive of recurrent neural networks (RNNs), while the bounds on the output are reminiscent of logistic regression. We expect the RNN to outperform and use the logistic regression as a benchmark.

For both models, we use 90% of the data as training and 10% for validation.

[7] `https://nlp.stanford.edu/projects/glove/`

II.1 **RNN**

We implement the RNN with Tensorflow and test the following set of hyperparameters.

Hyperparameter	Choices
Loss	Mean Squared Error
# of LSTM units	$\{64, 128, 256, 512\}$
Batch size	24
Learning rate	$\{0.001, 0.0005, 0.0001\}$
# of iterations	$\{100, 500, 1000, 2500, 10000\}$

Table 1: *These choices of hyperparameters will be visualized later.*

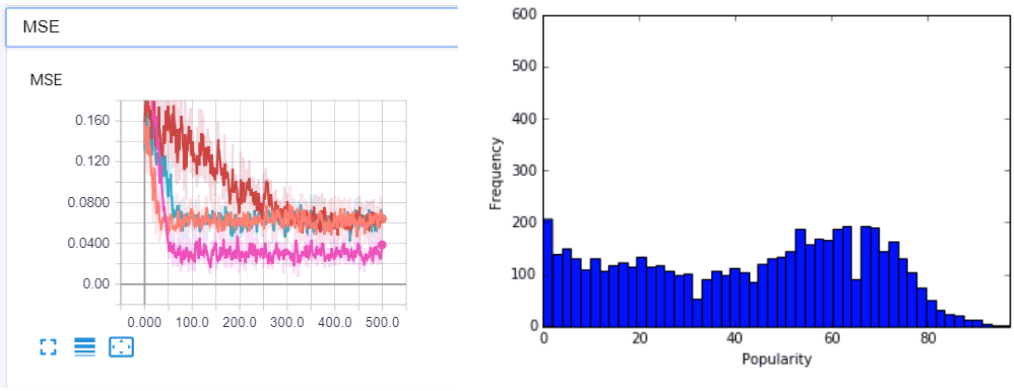


Figure 2: *Adjustment #1: unbiasing the data. The left depicts our initial Tensorboard visualization of RNN training. The blue curve converges to low error, but we found out that our data only contained popular songs since we crawled a list of popular artists; thus the error is artificially low from low data variability. After we sampled random songs to even out the popularity distribution, the curves converge to a very different accuracy. The right depicts the popularity histogram of the unbiased data.*

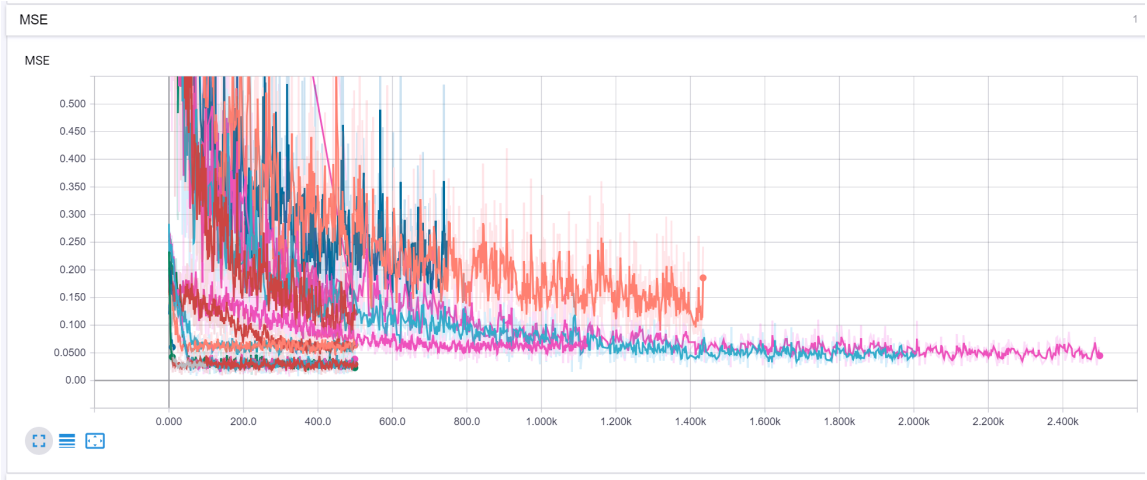


Figure 3: *Adjustment #2: hyperparameter optimization. The Tensorboard training curves show reasonable convergence for all choices of learning rate and architecture. We select a learning rate of 0.001 and 256 LSTM units.*

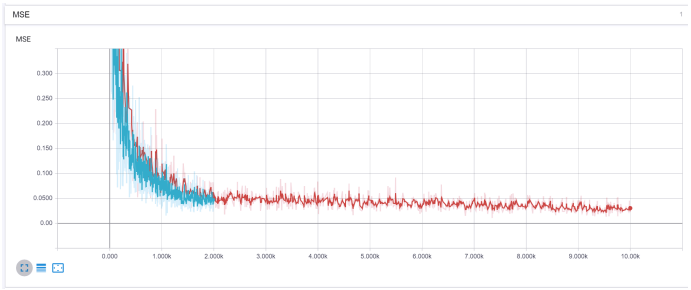


Figure 4: *Adjustment #3: early stopping. The Tensorboard training curves show a significant decrease in MSE loss between 2500 iterations and 10000 iterations. In reality, there is no significant difference in accuracy between the two (roughly 5.5% error for each). We implement early stopping for expedience and to avoid overfitting.*

Our final mean squared error on the testing data (10% of the original dataset) is 4.798%. In full disclosure, we were surprised that any learning over the structure of the lyrics was possible at all. In order to confirm that the model was not only learning over songs of a certain popularity range, we plotted the MSE against popularity (reminiscent of a residual plot).

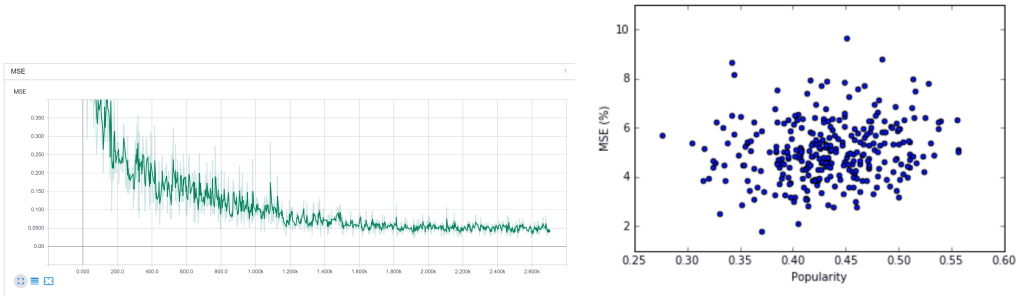


Figure 5: *Final model with error below 5%. Scatter demonstrates no significant bias.*

Finally, we select some songs and random text sources not found in the dataset to qualitatively evaluate the outputs of the model. We categorize the selected outputs based on whether they are above or below the mean output (0.401).[8]

Category	Sources	Popularity
Rightfully popular songs	Bohemian Rhapsody (Queen)	0.57
	Shape of You (Ed Sheeran)	0.45
	Never Gonna Give You Up (Rick Astley)	0.41
Undeservedly popular songs	Viva la Vida (Coldplay)	0.27
	Work (Rihanna)	0.29
	Mary Had a Little Lamb	0.25
Surprisingly good text	Gettysburg Address	0.41
Deservedly awful text	Neural net lecture notes	0.35
	Tragedy of Darth Plagueis the Wise	0.290
	Excerpt from "Twilight" by Stephenie Meyer	0.36

Table 2: *We see that the RNN model does distinguish on some dimension of quality, although the narrow range of its outputs ([0.25, 0.6]) fails to capture what should be a much higher variability ([0,1]).*

[8]These categorizations do not reflect the opinions of the authors.

II.2 **Logistic Regression**

First, we convert the featurized word vectors into a bag of words representation. We then perform the logistic regression by linearly regressing from the bag of words onto the logit of the popularity score, where the logit function is:

$$\text{logit}(x) = \log \frac{x}{1-x}$$

We use a learning rate of 0.001 and obtain a median MSE of 12%, which is significantly worse than the error of the RNN.

We conclude that the RNN models lyrics $\Longrightarrow$ popularity much better than the bag-of-words logistic regression.

VI. Conclusions

In trying to understand what makes music influential, we used song data scraped from the Genius API and Spotify API. We began with an exploratory data analysis through k-means clustering to understand aspects that make music enjoyable. In particular, the popularity, danceability, and energy metrics from the Spotify API were used, but no distinct correlation or useful clustering was found - thus, we simply chose to use popularity as a key metric in our analysis.

We then implemented a directed instrumental graph of the songs based on sampling references in order to measure Kleinberg centrality. Our findings are mostly in line with our expectations - older songs tend to have greater authority centrality (they are sampled by more songs) while more recent songs tend to have greater hub centrality (they sample more songs).

The final step of our analysis involved predicting track popularity based on lyrics. We decided that vectorizing the lyrics and using a RNN with Tensorflow was most appropriate. Our final RNN model achieved an out-of-sample MSE of 4.798%. This superb performance was robust across the entire spectrum of song popularity and outperformed the 12% MSE of the benchmark logistic regression model.

While we were able to build an accurate and robust model, we still encountered some limitations. The main limitation was that our RNN does not capture the full range of variability in popularity that our data contains. For future research, we would like to address this issue by expanding our dataset to include more songs that fall on the extremes of popularity. Furthermore, we would like to incorporate genre data by ensembling separate RNN models for different genres in order to potentially achieve higher accuracy.