

Gucci Gang and the Rise of Mumble Rap: an ensemble ANN-RNN approach to analyzing song lyrics

Cristobal Sciutto - *csciutto* - 06099425

Project Repository: github.com/csciutto/mfdoom

1 Introduction

With the recent surge in popularity of artists such as Lil Pump, Desiigner, and Lil Yachty, many "hip hop heads" have found themselves confused. Hip-hop has historically focused on spitting bars, with artist such as Eminem, Kendrick Lamar, and A Tribe Called Quest reigning as the greatest lyricists of all time. Nevertheless, as of late, the songs most played according to Billboards charts and Spotify rankings show very little lyrical prowess. Just look at the lyrics of "Gucci Gang", the current #3 song on the Hot 100 Billboard.

The original aim of the project was to incorporate a variety of facets of songs such as key singature, primary octaves, BPM, played instruments, etc. However, the complexity of extracting such features, which would ideally lead to MIDI files for each songs, led me to hone down on lyrics as the major area of study. Textual analysis is an area with wide literature, and as explained above, lyrics account for the major change of style as of late. In an attempt to understand this transition, I decided to attempt to use an ensemble of an ANN and RNN to predict the commercial success of tracks based on their lyrics.

2 Model and Algorithm

The task being tackled is predicting the popularity of hip-hop songs based on their lyrics, using Billboard charts as a heuristic for commercial success. Lyrical features of songs in the three major categories of semantics, structure, and vocabulary were extracted. The ideal end model that would be used is an ensemble of a recurrent neural-network (RNN) with a traditional neural network (ANN). The ANN would take into consideration features related to the totality of a song, such as lexical richness and song structure, while the RNN would be able to extrapolate on the sequentiality of lyrics, e.g. repeated catch lines.

In order to have a parameter of comparison for the performance of the predictor to be created, a baseline was determiend as the average score of all other songs by the same artist. For example, if Lil Uzi Vert were to release a new song, the baseline would average his existent scores in the database. A simple Linear Regression was used as a second baseline as a more complex approach.

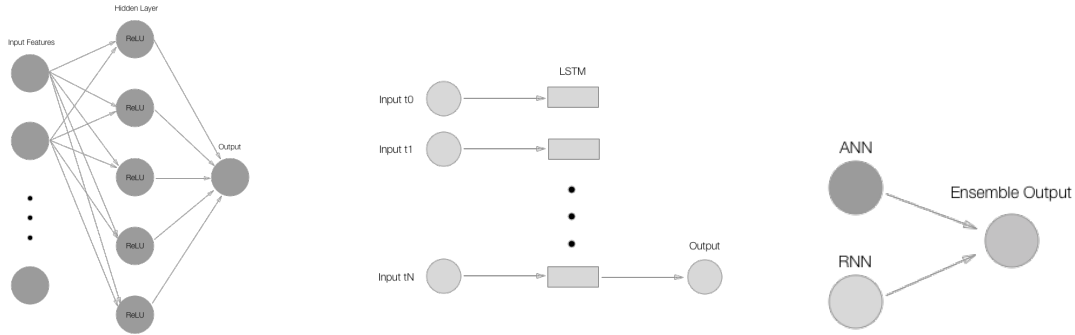


Figure 1: ANN, RNN, and Ensembled model respectively.

Then the complexity of the model was then progressively increased in order to account for correlations between features. Successive layers were added to the linear regression model. For the ANN, tests were run on different combinations of hyper-parameters and regularizations of weights to minimize both the mean squared error and the mean absolute error. Then, an RNN was trained on the sequential data. Finally, an ensemble composing the best ANN and RNN was to be used.

3 Feature Extraction

For each song, two distinct feature sets were created: one for the ANN and another for the RNN. The former is composed of general song features, involving the entire song as the atomic unit of measurement. These features can be split into three generic areas: semantics, song structure, and vocabulary. The latter has features of similar structure, however the atomic unit is a line of the song rather than its totality. This reduces the feature-set slightly as some features such as the number of stanzas, or the types of verses no longer make sense. A single song is consequently analyzed as a sequence, and it is from this characteristic that a recurrent neural-network gains power.

The key to the success of the predictor is making the features as expressive as possible, given the relatively small data set of Billboard charts since 1995, resulting in 4479 songs. *Rhyme and Style Features for Musical Genre Classification by Song Lyrics*¹ was a great starting point for some of the features. Furthermore, a lexicon mapping between words and emotions² (EmoLex) was used. The final features used were:

- Semantics
 - For each category in EmoLex, the proportion of words in each affect category.
 - For each part-of-speech in the NLTK universal tagset, the proportion of words.
- Structure
 - Length of song (# of words) and # of distinct words

- Number of stanzas and average length of stanzas (# of words)
- Number of lines of lyrics and average length of line (# of words)
- Vocabulary
 - Indicators for the n most popular words in song
 - Percentage of the song composed by common hip-hop songs, e.g. "money"
 - Percentage of informal words / sounds (e.g. "ooou")
 - Counts of !, ?, and numbers

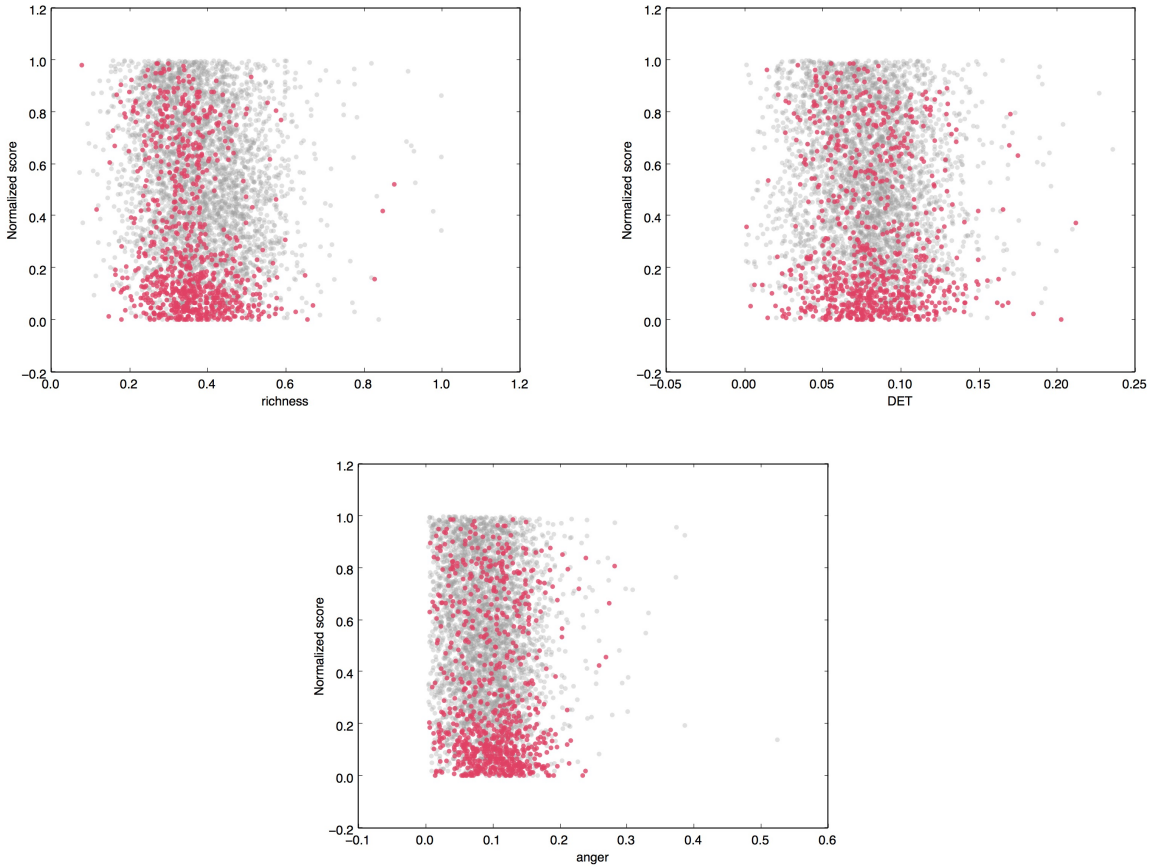


Figure 2: Distributions of **Richness**, **Determinants**, and **Anger**

Figure 2 above shows the distribution of three distinct features among the dataset. Data points in red correspond to the temporal split after 2015. It is important to note the near zero correlation between the features as indicative of there not being a formula for producing a Billboard hit. All 50 of the final features used for training have similar distributions, with the exception of those with a disproportionate amount of data points at the zero-value. This fact is further explored in the Discussion and Results section below.

These features are exemplified below for Gucci Gang, the trap song by Lil Pump, and m.A.A.d City by Kendrick Lamar.

	Gucci Gang	m.A.A.d City
Semantics		
- Distr. EmoLex cat, e.g. anger:	0.27	0.12
- Distr. NLTK POS universal, e.g. DET:	0.06	0.12
Structure		
- Distr. verse types, e.g. verse:	0.2	0.44
- # of stanzas:	6	10
- # of lines:	54	133
- Avg. lines / stanza:	71	113
- Avg. words / line:	7	8
Vocabulary		
- Total word count:	431	1133
- Non-word / total:	0.25	0.31
- Vocab. salience (if-idf):	0	0
- !-count:	1	17
- ?-count:	9	15
- #-count:	0	2
- % common words, e.g. love:	0.92	0.0

Scores are calculated proportional to the peak position of the song on the Billboard charts and the time spent on the charts. Since a lower peak position corresponds to a higher score, a simple paraboloid was used to map the position to a $[0, 1]$ scale, which was then multiplied by the number of time spent on the charts.

$$y_{raw} = r_b \cdot t_b = \frac{(101 - peakPos)^2}{10000} \cdot t_b$$

Consequently, the highest score would be achieved by a song which reached the number one spot, while staying on the charts for a significant amount of time. Using the peak position rather than the average position, along with the time on charts, compensates for songs that did not achieve a top spot in a given week due to increased competition e.g. several top songs being released in the same week.

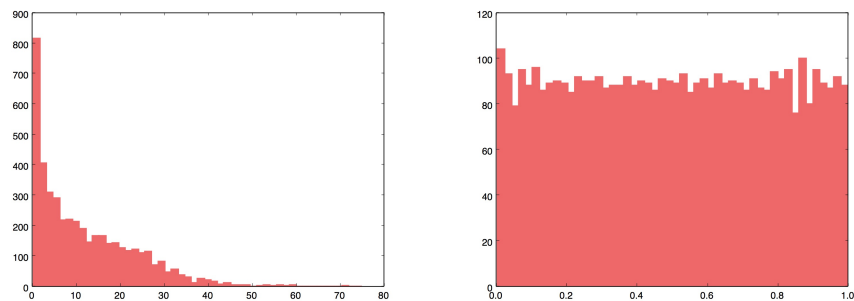


Figure 3: Score distributions before and after percentile normalization

The number computed is then percentiled in order to evenly distribute values. Note that this imposes linearity in quality of songs, which while not necessarily true, is necessary in order to avoid over-fitting on the outliers in the upper range.

The datasets were split into training, validation, and testing sets. Training sets were used to perform the gradient descent which optimizes weights for each of the models. The validation sets were used to optimize hyperparameters of models such as number of hidden nodes in the ANN, or the number of epochs to train for in order to avoid over-fitting. Finally, the testing sets serve as a final value to evaluate the model on.

This split was done in two ways, in order to tests for distinct theses. The first split was randomized, with 60% of the data used for training, 24% for validation, and 16% for testing. The second split was temporal. Songs released starting in 2015 were used as the testing set, while a randomized 70% of the rest of the songs were used train, and 30% to validate. In theory, a worse performance on the temporal split than the randomized split would indicate that there has indeed been a significant change in lyrics, while the converse would indicate that no substantial change has occurred.

4 Infrastructure and Development

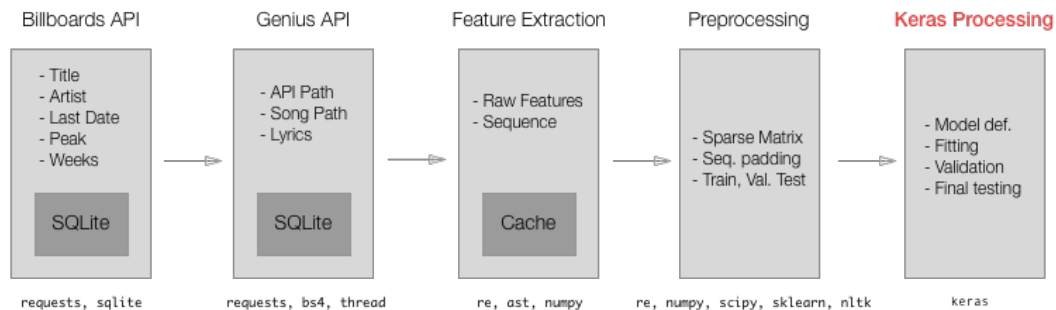


Figure 4: Data pipeline

The data pipeline followed is roughly illustrated above. Summarized, data was scraped from both the Billboard charts and Genius lyrics website. After tokenization and feature extraction, the raw features were cached. Then, when training, a preprocessing phase occurs in which the data is manipulated to correctly satisfy the Keras model. For recurrent training, sequences of fixed length must be built. The first 200 lines of songs were considered, and those who didn't reach the limit were padded. From there Keras with a TensorFlow backend was used to implement and train the models.

The code for this project is available at github.com/csciutto/mfdoom. The readme file explains the workflow in some detail.

5 Discussion and Results

For the final models trained, the results for the temporal and randomized dataset splits are below. Unfortunately, all techniques attempted failed at properly predicting the success of lyrics, leading to the abandonment of the final ensemble approach. For comparison, a 0.25 MAE is equivalent to always guessing the mean, and we are failing to breach this minimal threshold. Noticeably, all models results in nearly identical errors for the test set, a clear sign of underfitting. The baseline of merely averaging all other songs by the artist results in a better 0.27 MAE.

	TEMPORAL		RANDOM	
	MSE	MAE	MSE	MAE
LR.	0.13	0.32	0.10	0.30
ANN.	0.13	0.32	0.10	0.29
DNN.	0.13	0.33	0.11	0.30
RNN.	0.13	0.33	0.10	0.28

Throughout the development of this project, multiple iterations were made to try to combat this problem to little avail. The initial runs analyzed only songs from 2008 onwards, around 1300 total, leading to similar results as those above. This led to the hypothesis that a lack of data was the cause for the underfit. Furthermore, the initial featuresets were extremely sparse. The inclusion of the indicators for the most popular songs led to a dimensionality of 580, most of which were zero values. The immense size of the vectors compared to the 1300 samples was another strong indicator that underfitting was the culprit of the failures.

To fix these initial problems, more data was extracted. I incorporated both the Billboards Hip-hop R&B charts, and the Billboards Rap charts, spanning from 2000 onwards. This would account for roughly 3000 songs. Furthermore, popular songs was removed and other features were incorporated such as vocabulary salience through TF-IDF analysis, parts of speech, and counts of common words. This reduced the dimensionality of the feature vectors to 50 for the NN while increasing expressivity. Nevertheless, this led to negligible improvements. The dataset was further increased to 1995 onwards (4479 songs), and parsing of words was made more restrictive with simpler regular expressions to eliminate non-alphanumeric symbols from the raw lyrics.

There are two main explanations for the failure: underfitting or randomness. Our dataset is a mere 4500 samples, of which only around 3000 are used to train. Furthermore, the lyrical domain is not the sole domain related to music, and therefore cannot fully express a song. This makes underfitting likely, yet potentially solvable. Nevertheless, there might simply be no correlation between song features and their commercial success, which seems to be justified by the feature distribution shown above in Figure 2. Both Eminem and Future have had songs top the Billboard charts, despite extremely different styles (and critical acclaim). It is possible that we can't reduce down music to an array of features.

It is also very important to acknowledge two central problems in the dataset: lack of songs

outside of the Billboard charts and the improper English used in songs. Due to the difficulty of getting a well-distributed set of songs outside of the Billboards to counter-balance songs in the charts, I ended up only analyzing the relative success of songs, given that they had made the charts. For an interesting analysis of the probability a song makes the charts given their lyrics, another CS 221 project is cited below in Relevant Literature. The second problem stated relates to the dependence on clean lyrics for some of the features used. For example, the EmoLex only identifies the word "hundred", while a rapper might write "hunnid". This leads to sparser data and consequently underfitting.

For future developments, it will be interesting to incorporate an even larger number of features, even outside of the lyrical domain, e.g. rhymes, bpm, instrumentation. Additional songs outside of the Billboard charts must be incorporated to the dataset to eliminate any unwanted bias. Furthermore, a detailed parsing of lyrics would reduce the sparsity of data, possibly combating the underfitting.

6 Acknowledgements

I would like to express gratitude to my CA Michelle Mei for her feedback on the previous phases of the development of this project. Furthermore, I would like to thank my roommate Vishnu Sarukkai for helping identify core problems and proposing solutions.

This project would not have been possible without several open-source tools available: Keras, Scikit-Learn, Numpy, guoguo12's Billboard API, and the Genius API

7 Relevant Literature

1. Rudolf Mayer, Robert Neumayer, Andreas Rauber. *Rhyme and Style Features for Musical Genre Classification by Song Lyrics*. ISMIR 2008.
2. Saif M. Mohammad and Peter D. Turney. *Crowdsourcing a Word-Emotion Association Lexicon*. Arxiv 1308.6297
3. Doug. *How to choose the number of hidden layers and nodes in a feedforward neural network?* Stack Overflow
4. Sandip Srinivas, Omar Solis, Neel Sesh Ramachandran. *Predicting Billboard Rap Songs Based on Lyrics*. CS 221 Project Fall 2017.