Whose Rap is it Anyways?

Using Machine Learning to Determine Hip-Hop Artists from their Vocabulary

Alex Wang, Robin Cheong, Vince Ranganathan
{jwang98, robinc20, akranga}@stanford.edu

December 14, 2017

Abstract

This paper presents results of using supervised classification models to predict the authorship of rap lyrics based solely on the vocabulary. The models were trained on approximately two thousand songs by twelve artists. The strongest resulting model is capable of classifying the original artist correctly 74% of the time on unseen lyrics.

1 Introduction

Authorship identification and verification is a task that is becoming increasingly relevant due to machine learning. The problem is to select the true author from a given set of possibilities, based on characteristics of the text. Current algorithms exist to identify authors of literary prose, news articles, and tweets using tools such as natural language processing, support vector machines, and deep learning.

Rap, however, differs from prose in many respects. Rap songs are typically around 400–600 words in length, which means that there is a very limited amount of information off of which to make a prediction. Much of rap music also focuses around similar themes, one of the reasons why guessing rap authorship based on language alone is a difficult task.

Our task is to **identify the rapper of an input song's lyrics from a given set of rappers**. The input to our algorithm is the lyrics of a song, represented as a single string. This is passed to the classification algorithm – any of softmax regression, naive Bayes, a neural network, or a set of SVMs – that outputs the its prediction for the original author. The scope of the project is currently limited to a fixed *twelve* rappers, spanning three decades of rap music, that have "sufficiently extensive discographies" (four or more studio albums or mixtapes).[1]

Part of our motivation for undertaking this exploration is that we are interested in using error analysis to reveal and explore fascinating and non-obvious characteristics about rap lyrics: What are common and recurring themes in rap music? What makes each rappers' lyrics special and unique, or in other words, which features does the algorithm choose to weight most heavily? We hope that results from this research can be used to shed light on these questions.

2 Related Work

There is fair amount of related literature non-specific to rap music. Stanko et al. [SLH13] implement a series of SVM one-versus-all binary classifiers to predict authors from novels, using features concerning the structure of their writing. An upside of this is that it takes advantage of the structural intricacies of the language (e.g. "occurence of conjunctions per sentence"), but it discards the actual vocabulary. The problem is approached from a different angle by Qian et al. [QHZ], who use and fine-tune a variety of deep learning techniques, presenting approaches to handling text at both a sentence and an article level. This approach takes the vocabulary into account, but is not concerned with the structure of the prose.

Mechti et al. [MJB] target a broader goal, using author profiling to classify authors of anonymous tweets and other social media-related forms of text. This is a notably different, and much more general, task than author identification, and is based on implicit information about the authors from the text.

Mohsen et al. [MEMG] implement deep learning techniques to develop characterizations of texts that can effectively capture differences between authors' writing styles, focusing primarily on the process of feature extraction. Stańczyk and Cyran [SC07] provide a proof-of-concept that neural networks are efficient for tackling problems concerning stylometric analysis. These papers present optimistic preliminary results, but these successes are based on models trained over

[1] The selected rappers are Nicki Minaj, 50 Cent, Snoop Dogg, Lil Wayne, Jay Z, Nas, 2Pac, J. Cole, Kendrick Lamar, Drake, Kanye West, and Eminem. The dataset comprises the entire discographies of all of these artists.

very large datasets.

There is little existing literature regarding human-level performance in author – let alone *rapper* – identification. This is likely due to the fact that interest in authorship identification has only risen noticeably in recent years with advancements in machine learning techniques.

3 Dataset and Features

3.1 Data Collection & Pre-processing

We used the Genius API, PyGenius, and BeautifulSoup to find the list of songs of each artists and scrape the lyrics for each of these songs off of Genius (genius.com).

During pre-processing, we (i) *removed all non-lyrical data including punctuation* from the lyrics. This is to avoid strings such as "cat", "cat,", and "cat?" from being treated differently, and to prevent indicators such as "[Verse 2]" from appearing. We (ii) *applied stemming* to each word using the Natural Language Toolkit [BLK09], since some words, especially verbs, have many different endings even though they be used in the same way. As an example, "walk", "walks", and "walked" all refer to the same action of walking. These new objects are hereafter refered to as *tokens*.

We noted that neither of these actions – punctuation removal and word stemming – affected words or sounds that are included in the lyrics but not present in the English language (or are considered 'explicit'). This is important, as we would not want pre-processing to affect the validity of our results in any manner.

We partitioned the dataset into train, dev, and test sets encompassing 70, 15, and 15 percent of all songs respectively. These songs are split up randomly *for each artist* to ensure that there is balance across the three sets.

3.2 Feature Extraction

The features are determined by the **vocabulary** of a song, which is the words that it uses. We used four metrics of 'term frequency' (TF) [Ull11] that determine the importance of a word in a song:

count	number of times word w occurs in the song, denoted $f(w)$
binary	1 if word w occurs in the song, else 0
log	$1 + \log f(w)$, or 0 if $f(w) = 0$
norm	$\frac{1}{2}\left(1 + \frac{f(w)}{\max_{w'} f(w')}\right)$, preventing bias towards lengthier songs

Thus for every input, we have four potential feature vectors, depending on the specific TF metric chosen for the experiment.

These values are multiplied by the IDF (Inverse Document Frequency) of the token, to weight words based on how frequently they appeared across all documents.

$$\text{IDF}(w) = \log \frac{m}{\sum_{i=1}^{m} \mathbf{1}\{w \in x^{(i)}\}} \tag{1}$$

Thus, extremely common words like 'the' and 'is' would have a lower weight than uncommon words like 'purple'.

During the feature extraction process, we construct the model's "Bag of Words" (or "lexicon"), which is the union of the set of tokens present in each song. Any token that occurred fewer than 10 or more than 1,000 times across the data was filtered out to reduce overfitting to rare words (which are strongly indicative within the training set, but not generalizable), thereby decrease the total number of features from 29,620 to 4,970. We noted that this filtering process increased the accuracy of each of our models.

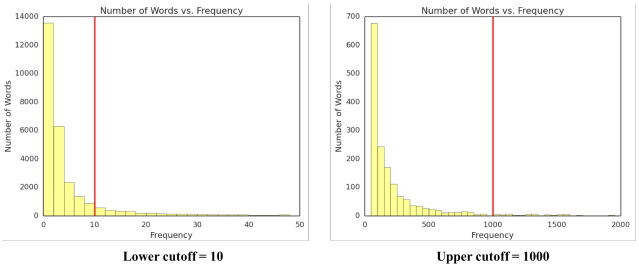

Figure 1: Lower and upper cutoffs based on word frequency in the training set.

Thus, our dataset is given by $\{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$ where $x^{(i)} \in \mathbb{R}^n$ and $y^{(i)} \in \mathbb{R}^K$, where n is the number of words in the vocabulary and $K = 12$ is the number of classes that we are classifying.

4 Methods

We proposed and developed the following models:

4.1 Naive Bayes

As a simple baseline performance measure, we generalized both the **multi-variate Bernoulli event model** and **multinomial event model** of Naive Bayes to a multi-class classification setting and added Laplace smoothing. Thus, our MLE parameters were given by:

$$\phi_{j|y=k} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = k\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = k\} + K} \tag{2}$$

$$\phi_k = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = k\}}{m} \tag{3}$$

for the multi-variate model and

$$\phi_{j|y=k} = \frac{\sum_{i=1}^{m} x_j^{(i)} 1\{y^{(i)} = k\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = k\} + K} \tag{4}$$

$$\phi_k = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = k\}}{m} \tag{5}$$

for the multinomial model.
Both estimate the probabilities of observing feature j given that example i is in class k by summing either the appearance or the number of occurrences of the feature given a class (multi-variate vs. multinomial) and normalizing over the number of examples in class k. The probability of a class ϕ_k is just the number of occurrences of the class in the data set. We can estimate the likelihood of observing a new data point x then by calculating

$$p(x) = p(y) \prod_{i=1}^{n} p(x_i|y) \tag{6}$$

and make a prediction for its label by finding the class k that maximizes the log-probability of the data:

$$\underset{k}{\text{argmax}} \sum_{i=1}^{n} \log[p(x_i|y=k)] + \log[p(y=k)] \tag{7}$$

We also note that Naive Bayes relies on the counts or appearances of individual words to find the maximum likelihood estimation of the parameters, and so a log or norm representation of the features would not applicable for this model.

4.2 Softmax Regression

To improve from Naive Bayes, we implemented softmax regression which assumes that the likelihood of the data is given by

$$\sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{\exp\left(\theta_l^T \phi(x^{(i)})\right)}{\sum_{j=1}^{k} \exp\left(\theta_j^T \phi(x^{(i)})\right)} \right)^{\mathbf{1}\{y^{(i)}=l\}} \tag{8}$$

We trained the algorithm using batch gradient descent with a regularized version of the cross-entropy loss:

$$CE = -\sum_{i=1}^{m} \sum_{j=1}^{K} \mathbf{1}\{y^{(i)} = j\} \log(\hat{y}_j^{(i)}) + \frac{\lambda}{2} ||W||_2^2 \tag{9}$$

where $\hat{y}_j^{(i)}$ is the softmax output on example i for class j and λ was set to 1.

4.3 Neural Network

Compared to softmax regression, neural networks have the ability to capture relationships between features. By implementing a simple neural network, we thought it may pick up on a deeper structure within the lyrics and thus generalize well to the dev set.

Our final implementation is a single fully-connected hidden layer with 360 neurons. The hidden layer uses ReLU activation, and the output layer uses softmax activation for multiclass classification. To train our model, we used AdamOptimizer in Tensorflow with the Softmax Cross-Entropy loss function [AAB+15].

To make a prediction, we run an iteration of forward-propagation according to the formula:

$$z^{[l]} = W^{[l]} a^{[l-1]} b^{[l]} \tag{10}$$

$$a^{[l]} = g^{[l]}(z^{[l]}) \tag{11}$$

where $g^{[1]}$ is the ReLU function, and $g^{[2]}$ is the softmax activation function. We then take the argmax of the softmax output which corresponds to finding the most likely label. To update the set of weights, $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$, we perform back-propagation by computing the gradients of the cross entropy loss with respect to each of the layers.

4.4 Support Vector Machines

The last model we implemented is a One-vs-rest SVM classifier. We trained 12 different SVM classifiers on each of the authors, and predicted the artist that corresponded to the highest positive margins. Each individual SVM used the Gaussian kernel and hinge-loss function and was trained with scikit [PVG+11].

$$\ell_{\text{hinge}} = \sum_{i=1}^{m} \max(0, 1 - y^{(i)} \; W \cdot \phi(x^{(i)})) \tag{12}$$

To make a prediction, we input the test set's features into each SVM. We then obtain each SVM's prediction and corresponding margin, and find the argmax of product of the two values.

$$\hat{y}^{(i)} = \underset{k}{\text{argmax}} \left(\hat{y}_k W_k \cdot \phi(x^{(i)}) \right) \tag{13}$$

5 Experiment, Results and Discussion

We chose accuracy as our primary metric for success. The accuracy levels for each model-feature type pair, over the train and dev sets, are as follows:

	NB	Softmax	NN	SVM
Binary	90.7	90.7	98.6	97.0
	69.0	69.1	66.5	66.3
Counts	96.0	99.0	99.1	98.4
	69.1	66.9	65.5	66.5
Log		98.7	98.5	98.5
		71.9	59.7	71.9
Norm		98.0	99.1	**97.8**
		74.1	72.7	**77.0**

Figure 2: Accuracy levels for each model. White represents training set; light blue represents dev set; red is best model.

The best performing model is the SVM using the TF = Norm feature vector, and after retraining on a combined train-dev set, we get a test accuracy of 74.3%.

Below are the receiver operator characteristic (ROC) graphs illustrating the diagnostic ability of each

of the twelve SVMs (split between two graphs for readability purposes).

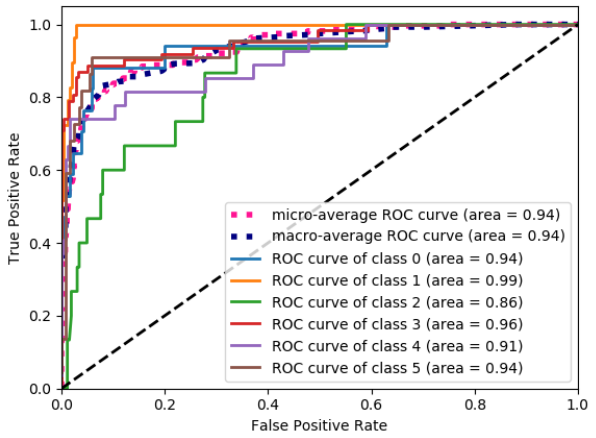

Figure 3: ROC graph for classes 0 through 5

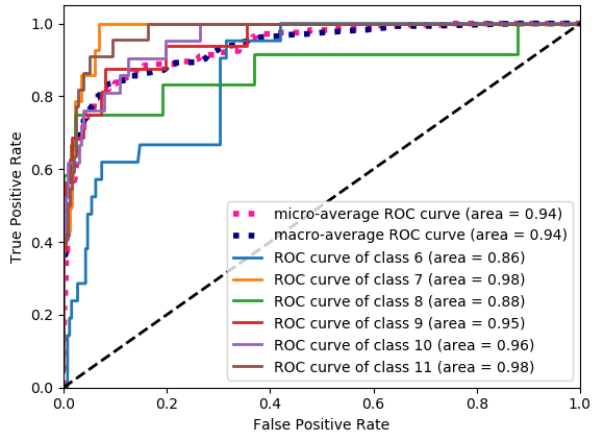

Figure 4: ROC graph for classes 6 through 11

As evident from the ROC graphs, our One-vs-Rest SVM classifier does very well, achieving high true positive rates and low false positive rates. Similarly, the AUROC (area under the ROC curve) metrics are all relatively close to 1.0.

		Predicted												
		Nicki Minaj	50 Cent	Snoop Dogg	Lil Wayne	Jay Z	Nas	2Pac	Jcole	Kendrick Lamar	Drake	Kanye West	Eminem	Recall
Actual	Nicki Minaj	**13**	0	0	1	0	0	1	0	0	1	1	0	0.764706
	50 Cent	0	**29**	0	0	0	0	0	0	0	0	0	0	1
	Snoop Dogg	0	0	**4**	0	1	3	7	0	0	0	0	0	0.266667
	Lil Wayne	0	0	1	**56**	1	0	0	0	0	2	2	0	0.903226
	Jay Z	0	0	0	2	**19**	1	0	2	1	0	2	0	0.703704
	Nas	0	1	0	1	0	**19**	0	0	1	0	0	0	0.863636
	2Pac	0	0	5	0	1	2	**12**	0	0	0	0	1	0.571429
	Jcole	1	0	0	1	0	0	0	**12**	0	0	0	0	0.857143
	Kendrick Lamar	1	0	0	1	1	0	0	1	**7**	0	1	0	0.583333
	Drake	1	0	0	1	0	0	0	0	0	**12**	2	0	0.75
	Kanye West	1	2	0	0	2	1	0	0	0	0	**15**	0	0.714286
	Eminem	0	1	1	1	0	0	0	1	0	2	0	**16**	0.727273
	Precision	0.76470588	0.8788	0.363636364	0.875	0.76	0.73	0.6	0.75	0.777777778	0.706	0.652173913	0.941176	

Figure 5: Confusion matrix for the SVMs-Norm model

From the confusion matrix, we can see that the classifier does poorly on Snoop Dogg, 2Pac, and Kendrick Lamar. What is interesting is that the classifier seems to have trouble discerning between Snoop Dogg and 2Pac, often predicting one person's song as the other's. One possible reason for this phenomenon is the fact that both started their careers in the early 90's and were very productive in the mid 90's. This notion is further supported by the fact that the second most guessed artist for Snoop Dogg's songs is Nas, who was also very prominent in the 90's compared to the other rappers in the list. Perhaps time period plays a very important role in the word choice and style of rap songs.

5.1 Hyperparameters and Optimization

5.1.1 Softmax Regression

After trying a few different learning rates, we found that using $\alpha = .002$ as the initial learning rate was effective. Once the change in the L2-norm of the weights after each iteration was less than .01, however, we used

$$\alpha^{(t)} = \frac{||W^{(t)} - W^{(t-1)}||_2^2}{10} \tag{14}$$

so that the learning rate would shrink as the algorithm progressed.

Because the number of songs we had per artist was relatively small, we decided to use *batch gradient descent* instead of mini-batch or stochastic gradient descent, and set our convergence criteria to

$$||W^{(t)} - W^{(t-1)}||_2^2 < \epsilon \tag{15}$$

where an ϵ value of 10^{-5} was found to work best. Later, early stopping was used to reduce overfitting with the maximum number of iterations set to 400.

5.1.2 Neural Network

We experimented with several setups, each with different numbers of layers, numbers of hidden nodes per layer, and regularization constants. We identified that the neural network was very prone to over-fitting, and additional layers exacerbated this issue. The best results were achieved with a single hidden layer with 360 neurons, and a regularization constant of $\lambda = 0.05$ allowed the model to generalize well while maintaining high accuracy on the training set.

5.2 Sources of Error

It is particularly important to understand the likely sources of error, as none of the models performed exceedingly well after extensive data pre-processing and hyperparameter tuning. We have identified the following as key sources of error:

1. **Collaborations between rappers**: artists frequently feature other lyricists on their songs. As a result, it is possible that *as much as half* of the lyrics to a song are not written by (and therefore should not be attributed to) the original artist. To adjust for this, the model can be altered to *report the two most likely artists*. For each of softmax, neural networks, and SVMs (all with Norm feature vectors), the accuracy of the top two guesses is as follows: softmax: 88.1%, neural networks: 87.6%, SVMs: 88.3% .

2. **Ratio of #features to #training examples**: with a (post-filtering) feature vector size of 4,970 and a training set size of ≈ 200 for each rapper, we have $m \ll n$. This contributes to the observed issue of high variance, since the models become much more prone to overfitting to the training data, which we confirmed from plotting the learning curves on the data:

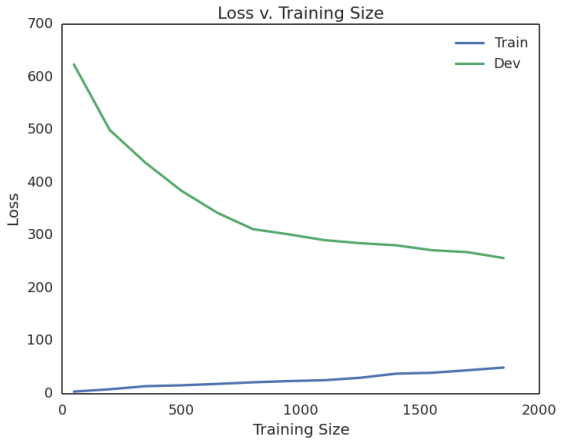

Figure 5: Plot of Loss vs. Training set size for the one-vs-all SVMs

Due to the nature of the data–i.e. since rappers have a limited number of published works–it is not possible to significantly increase m. However, it may be possible to further reduce n through a feature-selection process such as PCA.

To address this issue, we attempted to add regularization, but identified that regularizing the weights did little to prevent overfitting. In fact, for small values of λ ($\lambda < 5$), there was no noticeable change in train and dev accuracy, and for large values of λ, train and dev accuracy decreased noticeably. Upon examining the weight matrix, we discovered that the learned weights without regularization were already small. Thus, regularization had little effect.

We also implemented early stopping techniques and found that, rather than allowing the algorithm run until convergence, restricting the number of iterations to 300 slightly improved dev set accuracy of the softmax model from 74.1 to 74.4.

3. **Incomplete lexicon**: the model's "bag of words" is determined and fixed during the training period. As a result, any tokens identified in the test input that were not found during the training phase are ignored entirely. This means that the algorithm is discarding a significant amount of valuable information. One approach to resolving this is using a model that uses a "similarity" metric to compare unseen tokens to known ones. This for example, might connect the previously-unseen word "cash" to the known word "money", and affect the corresponding feature component.

4. **Limitations of the "bag of words" model**: the "bag of words" model entirely discards the structure of the song and relationships between different words, which removes a huge component of an artist's style that would be useful in identifying one from another. To bypass this limitation, an algorithm that takes into account the entire structure of the data, such as an RNN, could be used.

6 Conclusion and Future Work

We have developed and tested multiple classification models to identify the author of an input of rap lyrics based on vocabulary. Hyperparameters were tuned in order to filter out extremely rare or common words from consideration, and to experiment with different types of feature extractors. The highest performing model was the set of one-vs-all SVMs, using the Norm feature extractor. This achieved a combined train-dev set accuracy of 96.8% and test set accuracy of 74.3%. However, in general, all algorithms aside from Naive Bayes performed in the mid-70s in terms of accuracy (softmax: 74.1%, NN: 72.7%). Interestingly, neural network's performed the worst, likely because, as a more complex model, it easily overfits to the training data and fails to generalize well.

Future work on this topic would primarily involve reducing the variance of the model by modifying the dataset to separate collaborative songs into independent contributing artists, reducing the number of features, using measures of word similarity, and implementing k-fold cross validation.

Contributions

- Alex Wang: data collection, testing, Naive Bayes, neural network, SVMs, documentation
- Robin Cheong: softmax regression, testing, feature extraction, Naive Bayes, error analysis, documentation
- Vince Ranganathan: data pre-processing, feature extraction, softmax regression, error analysis, documentation

References

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015, Software available from tensorflow.org.

[BLK09] Steven Bird, Edward Loper, and Ewan Klein, *Natural Language Processing with Python*, O'Reilly Media Inc., 2009.

[MEMG] Ahmed M. Mohsen, Nagwa M. El-Makky, and Nagia Ghanem, *Author Identificaton using Deep Learning.*

[MJB] Seifeddine Mechti, Maher Jaoua, and Lamia Hadrich Belguith, *Machine learning for classifying authors of anonymous tweets, blogs, reviews and social media.*

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[QHZ] Chen Qian, Tianchang He, and Rao Zhang, *Deep Learning based Author Identification.*

[SC07] Urszula Stańczyk and Krzysztof A Cyran, *Machine learning approach to authorship attribution of literary texts*, International Journal of Applied Mathematics and Informatics **1** (2007), no. 4, 151–158.

[SLH13] Sean Stanko, Devin Lu, and Irving Hsu, *Whose Book is it Anyway? Using Machine Learning to Identify the Author of Unknown Texts.*

[Ull11] Jeffrey D. Ullman, *Data Mining*, 2011.