# Artist Attribution via Song Lyrics

Michael Mara

December 12, 2014

## 1 Introduction

Song lyrics, separated from the audio signal of their song, still contain a significant amount of information. Mood and meaning can still be conveyed effectively by a pure textual representation. There has even been somewhat successful previous work on genre classification from song lyrics[7]. Building on previous work, we seek to build an artist attribution system for song lyrics.

This task is in the same vein as classic author attribution tasks, which often are trained and evaluated on extremely large datasets[8]; providing more data per author than it is possible to get for most songwriters. In order to focus our task, we focus only on rap, as the songwriter and performer are usually the same, and there is a heavy emphasis on distinctive forms of lyricism. We actually enshrine that first assumption in our statement of the classification task: given a textual representation of the lyrics of a rap, return the name of the *artist* who raps it. This is a limitation we will have to live with for now, there does not exist any large public database that provides ghostwriting information for rappers.

Potential use cases for such a classifier would be for detecting misattributed songs in a music library or as part of an auto-tagger in a music management system, along with other uses of author-attribution systems. A lyric-only classifier could also be used in an ensemble method that includes audio-only classifiers.

Following previous work[2], we initially attempt to distinguish between 4 prolific rappers (Eminem, Nas, Jay Z, and Nicki Minaj) before expanding the classification task to encompass more artists, testing thoroughly on a 12-artist dataset, and eventually testing on over 300 rappers at once.

## 2 Dataset

Previous work has run into the issue that there appears to be no reliable large dataset of lyrics with author attribution[2], so we follow their lead in constructing our own dataset.

Song lyrics were obtained via the Genius API using the Ruby gem rapgenius.rb[1]. They were then processed using the Python Natural Language Toolkit (NLTK)[2]. For each artist dataset, we downloaded the lyrics to all available songs by each artist, and created an ad-hoc blacklisting mechanism in python to remove translated lyrics and non-songs (rap genius sometimes has transcripts from movies or interviews with the artist). We also initially excluded songs that featured other artists even if our target artist was the primary artist on the track, in order to mitigate corrupted data from verses from the featured artist, obtaining the 4-artist (initial) dataset, consisting of 508 songs. Upon examination of the learning curve from the dataset, which suggested more data would give a significant benefit, we relaxed the requirement; then obtaining the 4-artist (extended) dataset (887 Songs) and the 12-artist dataset (2204 Songs).

As a final test, we also tried testing on a dataset made of all songs by all artists appearing on Wikipedia's List of Hip-Hop Musicians[3] with over 40 songs available on Genius. This resulted in a dataset with 348 artists (34,352 songs). Table 2 summarizes our three datasets.

## 3 Features and Preprocessing

Given the raw lyrics to a song, we first filter out song descriptors (such as "[Chorus]", "[Verse]") via a simple handcrafted regex, then tokenize the remaining lyrics. All features are extracted from this tokenized representation. Except in our final experiments, we stick to a simple bag-of-words model, which has proven to work very well on related tasks[8], often beating painstakingly handcrafted features.

In order to obtain the bag-of-words representation, we stem the tokens using the NLTK Snowball stemmer, construct a vocabulary consisting of every word

[1] `https://github.com/timrogers/rapgenius`

[2] `http://www.nltk.org/`

[3] `http://en.wikipedia.org/wiki/List_of_hip_hop_musicians`, accessed 12/10/2014

| Artist | Song Count |
|---|---|
| T.I. | 216 |
| 2Pac | 355 |
| Snoop Dogg | 304 |
| Ice Cube | 181 |
| Nelly | 114 |
| Lil Jon | 50 |
| Sir Mix-a-Lot | 59 |
| Ying Yang Twins | 38 |
| Eminem | 289 |
| Nas | 263 |
| Kanye West | 183 |
| Nicki Minaj | 152 |

Table 1: The artists and song counts from the 12-artist dataset. The 4-artist (extended) dataset consists of just the songs from the final four rows.

| Dataset | Song # | Vocab. Size |
|---|---|---|
| 4-artist (initial) | 508 | 3,439 |
| 4-artist (extended) | 887 | 5,101 |
| 12-artist | 2,204 | 7,977 |
| 348-artist | 34,352 | 33,031 |

Table 2: The artists and song counts from the 12-artist dataset. The 4-artist (extended) dataset consists of just the songs from the final four rows.

in the dataset, and construct a feature vector for each song consisting of the count of each instance of each word in the vocabulary appearing in the song. The resulting in a bag-of-words representation is ideal for our Naive Bayes classifier[9].

On top of this, we implemented two feature selection methods in order to hopefully improve generalization error[3]; first a simple document frequency thresholding, which removed words from the vocabulary if they did not appear in at least 5 songs. Second, we computed the $\chi^2$ statistic for feature selection[6].

$$\chi^2(w,a) = \sum_{e_w \in \{0,1\}} \sum_{e_a \in \{0,1\}} \frac{(N_{e_w e_a} - E_{e_w e_a})}{E_{e_w e_a}}$$

is computed for each artist/word pair, where $e_w$ is the occurence of the word $w$ (1 when it occurs, 0 when it does not), $e_a$ is the occurrence of the artist $a$, $N_{e_w e_a}$ is the observed frequency of co-occurence of the events and $E_{e_w e_a}$ is the expected frequency of the co-occurence of the two events if the two events were independent.

We then assigned a $\chi^2$ score to each word by taking the max over all $\chi^2$ from artist/word pairs involving the word:

$$\chi^2(w) = \max_a \chi^2(w,a)$$

We then chose $n$ features by choosing the $n$ words with the highest value of $\chi^2$.

Feature weighting in Naive Bayes with the Kullback-Leibler Measure[5] was briefly considered, but postponed due to the relative ineffectiveness of our initial feature selection methods.

In our final experiments, we add on part-of-speech (POS) bigrams to test the value of adding a proxy for syntactic structure. First each token in a song is converted into a POS tag using the NTLK, and then the count of each bigram of the resulting tags is used as a feature.

# 4 Models

We use our own MATLAB implementation of a multiclass Naive Bayes classifier using the multinomial event model and Laplace smoothing as our main model. This was chosen based on its widespread success in many text classification tasks.

As a sanity check, we also implement a model based on support vector machines. We use the built-in MATLAB fitcsvm() to train an ensemble of 1-vs-all binary SVM classifiers, on the same features used for Naive Bayes. We do multi-class classification by selecting the artist whose corresponding SVM returns the highest score. The default C parameter (known as the *BoxConstraint* parameter in some texts and the MATLAB documentation) causes severe overfitting (0 training error, 17%+ test error), so we also train with hand-tuned smaller C values.

# 5 Results

Note that all results use 10-fold cross validation unless otherwise specified. Taking a cue from Computer Vision (specifically the ImageNet classification tasks[4]), we report not only the standard error rate for our larger datasets, but also some of the Top-$N$ error rates, where an example is counted as misclassified if its correct label was not among the $N$ rated as most probable by the model. Note that the Top-1 error rate is identical to the standard error rate.

## 5.1 Initial Results

Our initial experiments used the 4-artist (initial) dataset, running the same classification task as Guo et al.[2]. With this dataset, and using all 3439 features, our Naive Bayes classifier achieves a test error

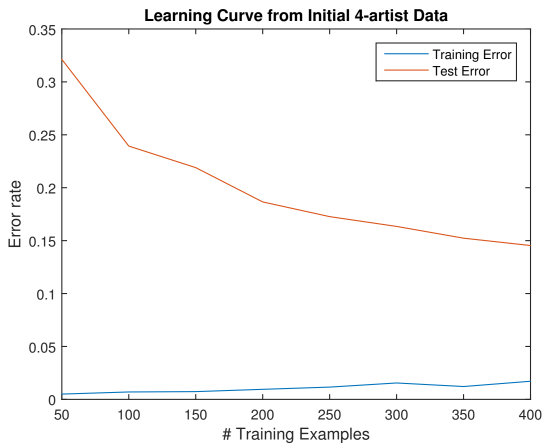



Figure 1: The learning curve from the initial data suggested more training examples would continue to decrease our test error rate.

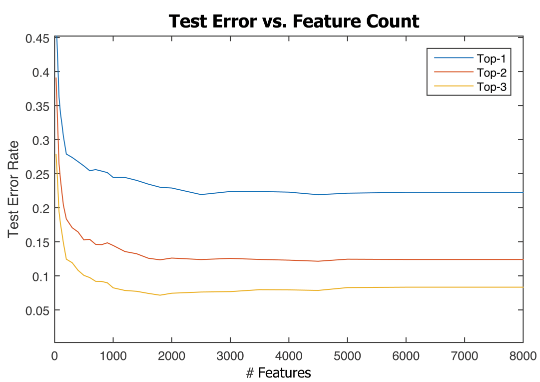



Figure 2: The test error rates for our Naive Bayes classifier on the 12-artist dataset as we very the number of features, using our $\chi^2$ selection criteria.

of 14.27%; slightly lower than Guo et al (at 15%). In order to improve results further, we plotted the learning curve (Figure 5.1), and saw that more training examples would likely benefit our model. This is when we got rid of the "no featured artists" constraint, and obtained our other datasets.

## 5.2 Adjusting Feature Count

In Figure 5.2, we show that our feature selection method does not seem to help results significantly, even on our 12-artist dataset, though it does allow the removal of many features without negatively affecting our error rates.

| Dataset | 4-artist (Initial) [508 Songs, 3439 Features] | | 4-artist (Extended) [887 Songs, 5101 Features] | | 12-artist [2204 Songs, 7977 Features] | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | Training Error | Test Error | Training Error | Test Error | Training Error | Test Error | Top-2 Test Error | Top-3 Test Error |
| SVM (C=1, n=2000) | 0 | 0.1880 | 0 | 0.1752 | 0 | 0.2865 | 0.1702 | 0.1141 |
| SVM (C=0.005, n=2000) | 0.018 | 0.1823 | 0.412 | 0.1909 | 0.0384 | 0.2336 | 0.1428 | 0.0991 |
| SVM (C=0.002, n=2000) | 0.421 | 0.2068 | 0.153 | 0.1788 | 0.0740 | 0.2508 | 0.1533 | 0.1141 |
| Naïve Bayes (All features) | 0.0195 | 0.1427 | 0.028 | 0.1244 | 0.0538 | 0.2227 | 0.1241 | 0.0834 |
| Naïve Bayes (n=2000) | 0.0199 | 0.1465 | 0.034 | 0.1226 | 0.0908 | 0.229 | 0.1262 | 0.0745 |
| Naïve Bayes (n=500) | 0.0489 | 0.1521 | 0.0696 | 0.1506 | 0.1811 | 0.2614 | 0.1528 | 0.1007 |

Table 3: Our results on our 3 main datasets using all of our models. Naive Bayes is the best model in all of our tests.

| Error Type | Error Rate |
|---|---|
| Training Error | 0.6395 |
| Test Error | 0.7901 |
| Top-3 Error | 0.7044 |
| Top-5 Error | 0.6642 |
| Top-10 Error | 0.5974 |
| Top-50 Error | 0.3608 |
| Top-100 Error | 0.2167 |

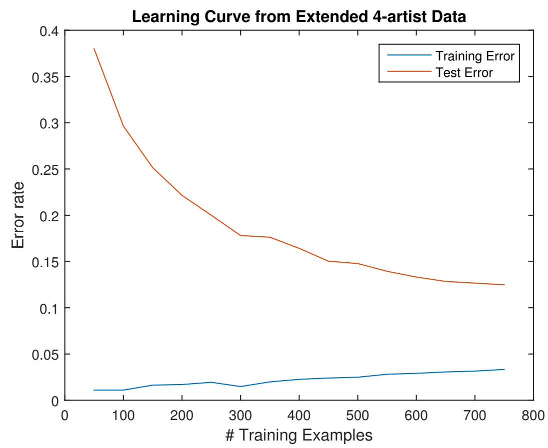
Table 4: Error rates for our Naive Bayes Classifier on the 348-artist dataset.

## 5.3 Main Results

Table 5.3 is a table of our main results, and Figure 5.3 gives the learning curves for the 4-artist (extended) and 12-artist datasets on our highest performing model. Figure 5.3 provides a visualization of the confusion matrix (averaged over the 10-fold cross-validation) of our highest performing model. The only anomalous result is the unusually poor classification of the Yin-Yang Twins, which our model hardly ever uses as the predicted label. This may be partially attributable to the fact that the Yin-Yang Twins have the lowest song count in all of our datasets at 38.

## 5.4 Scaling Up

In order to get a taste of how our best model performs on a dataset more than an order of magnitude larger (both in song count and artist count), we ran our Naive Bayes model on a dataset of 34,352 songs across 348 artists. Due to the fairly high error rate (though fairly low compared to chance), and high computational cost, we report results only for Naive Bayes using all available features (see Table 5.4).



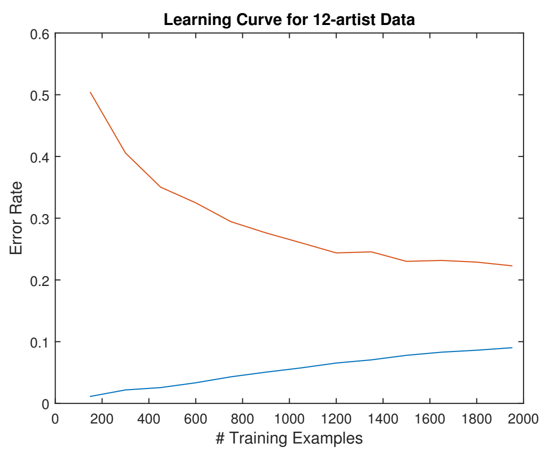




Figure 3: The learning curves for our Naive Bayes classifier using all available features on both the 4-artist (extended) and 12-artist datasets. Note the similarity to the learning curved for the 4-artist (initial) dataset.

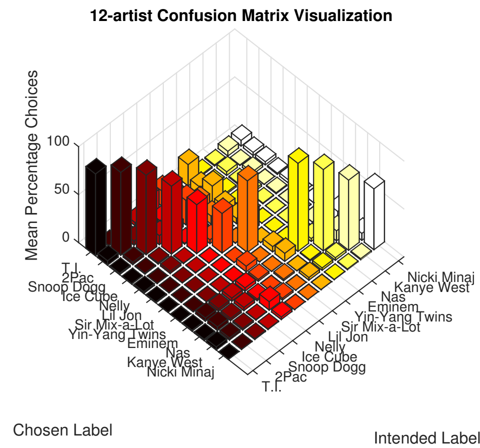



Figure 4: Confusion Matrix for our Naive Bayes Classifier on the 12-artist dataset (using all features). The one anomalous result is the high misclassification of the Yin-Yang Twins, who have the least number of songs in our dataset.

| Model | Bag-of-Words | + POS Bigrams |
|---|---|---|
| Naive Bayes | 0.1244 | 0.1382 |
| SVM (C=1) | 0.1781 | 0.1992 |
| SVM (C=0.005) | 0.1632 | 0.1799 |
| SVM (C=0.002) | 0.1767 | 0.1694 |

Table 5: Comparison between our base model and our model augmented with POS bigrams. Our tests showed no improvement (in fact a deterioration) from adding POS bigrams to our model. Although it improves the test error slightly on our SVM(C=0.002), the best error for the SVM still comes without the use of POS Bigrams.

### 5.5 Adding Features

As a quick final test, to try and get more information out of our limited number of training examples, we tried augmenting our bag-of-words model with part-of-speech (POS) bigrams, generated using the NLTK, as a proxy for local syntactic structure.

## 6 Discussion

Judging by Figure 5.2, our feature selection mechanism seem to be at best not-harmful; there's no noticeable improvement in the error rate by selecting smaller feature sets, though it also doesn't hurt until <2000 features on the 12-artist dataset. Examining our main results (Table 5.3) The Naive Bayes classifier does fairly well on the task, having a significantly lower error rate than the SVM, which takes far longer to train. Given previous successes using Naive Bayes, this is not entirely unexpected.

Our classification methods hold up fairly well even when tripling the number of categories, with our best classifier achieving <7.5% top-3 test error on the 12-artist dataset. We outperform Guo et al[2] on the 4-artist task, likely due to our larger quantity of training examples, via our programmatic data extraction.

Inspection of the learning curves (Figure 5.3) suggests more examples could lead to further reduction of our error rate, as training error is much lower than our test error. However, this is infeasible; artists only put out a finite number of songs, and most have a smaller discography than any of the twelve in our toughest classification task.

In order to get better classification, to the point where we could perhaps get reasonable classification on our 348-artist dataset, we must make better use of the current data. Raw bag-of-words, though elegant, throws away a lot of information that could be captured by higher-level features, such as rhyme scheme, sentiment, and syntactic construction. Although we found no benefit from adding POS bigrams, there are many other features that could be tried in a more extensive analysis, including rhyme and style features used in previous work[7].

# 7 Future

Future work should seek to extend the bag-of-words model, perhaps by implementing redundancy compensated bigrams[1], sentiment analysis, and rhyme and style features[7]. Class label reduction by using some clustering of artists could also prove useful (or perhaps necessary, as the sheer number of artists may drive the error rate far too high for even the most sophisticated of models).

This paper purposefully avoids the question of attributing ghost-written songs to songwriters, which could be an interesting challenge. Our datasets are also automatically generated from a crowd-sourced lyric repository; the error rate of our ground truth is unknown, and worthy of study in order to continue this line of research.

# 8 Acknowledgements

The author would like to thank Professor Ng and the CS229 course TAs for the fascinating introduction to machine learning over the course of the last few months.

# References

[1] Boulis, C., and Ostendorf., M. Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams. In *In Proc. of the FSDM* (2005).

[2] Guo, S., and Khamphoune, S. "i'm different, yeah i'm different": Classifying rap lyrics by artist. In *Stanford CS229 Final Project Papers* (2013).

[3] John, G. H., Kohavi, R., and Pfleger, K. Irrelevant features and the subset selection problem. In *MACHINE LEARNING: PROCEEDINGS OF THE ELEVENTH INTERNATIONAL* (1994), Morgan Kaufmann, pp. 121–129.

[4] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, p. 2012.

[5] Lee, C.-H., Gutierrez, F., and Dou, D. Calculating feature weights in naive bayes with kullback-leibler measure. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining* (Washington, DC, USA, 2011), ICDM '11, IEEE Computer Society, pp. 1146–1151.

[6] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[7] Mayer, R., Neumayer, R., and Rauber, A. Rhyme and style features for musical genre classification by song lyrics. In *In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)* (2008).

[8] Stamatatos, E. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol. 60*, 3 (Mar. 2009), 538–556.

[9] Telecommunications, V. M., and Metsis, V. Spam filtering with naive bayes – which naive bayes? In *Third Conference on Email and Anti-Spam (CEAS)* (2006).