Authorship Attribution of Song Lyrics

Abstract

Authorship attribution is a common application of forensic linguistics that can be performed on a variety of data types. The goal of authorship attribution is to predict the creator of a piece of linguistic data by analyzing the quirks and patterns of a text or audio sample and comparing them to a set of potential authors to determine the best match. In this paper, I apply this technique to a large database of song lyrics scraped from the Internet as I attempt to train a computational model to predict the performing artist of a given song. A key element of this project is to find a list of relevant features, or calculable information, that best distinguishes the songs of a certain artist from the songs of all other artists. For example, the most obvious difference between the line “In chilly sub-depth railways, the weathered concrete stairways provide me with a means of getting home” (from Owl City’s “Early Birdie”) and the line “So get out, get out, get out of my head / And fall into my arms instead” (from One Direction’s “One Thing”) is the presence of more unusual words in the former example than in the latter. Therefore, my model uses the inverse document frequency to determine the rareness of each word in the song and uses it to help find a matching artist. The entire feature set I discuss in this paper contains various types of linguistic information, although syntax is the most difficult to manage because the syntax of lyrics is strongly constrained by the meter of the song.
This topic is inherently susceptible to a data sparsity problem—the number of words in a single song may not be enough to effectively perform the statistical component of the model. In fact, the reason that I choose to define the author as the performing artist rather than the lyricist is that there is not enough lyricist information available. In many cases, the song’s metadata lists zero or multiple composers, both of which are incompatible with the machine learning algorithms I use from Python’s scikit-learn package. However, I claim that predicting the performing artist is still a worthwhile task because bands will choose to record songs that have similar styles—both in terms of the music and the lyrics.

Though my model does not correctly predict the majority of the artists, it does perform significantly better than chance, meaning that the selected features do give some indication of the performing artist. Although the success of the classifier is more visible with a smaller number of possible authors, the ratio between its accuracy and chance is maintained even when applied to a larger data set.

**Introduction**

Authorship attribution falls within the field of forensic linguistics, which deals with applying linguistic techniques to legal situations. In a courtroom, there is often an effort to match a piece of written evidence with the suspect responsible for it. For example, linguistic analysis can uncover the tone of a ransom note or verify the genuineness of a suicide note, adding crucial evidence to a trial (Olsson 2016: 2-3). This is one purpose of authorship attribution. However, this process is not constrained to the courtroom; it can be useful in other domains as well, such as plagiarism detection and
matching pen names to famous authors (Juola 2008: 250). For the purposes of this paper, I will focus on the application of forensic linguistics to song lyrics to show how similar success can be had in predicting authorship. I propose that this can be done by a computational system trained on a large lyric database with relevant features extracted. I will first summarize previous work in authorship attribution before describing some lyric-specific features I chose to incorporate into my classification model, explaining why they are appropriate for the task at hand. After noting additional improvements that could potentially improve the model’s accuracy, I will give a description of the computational techniques I used to generate the predictions.

The method by which I determine authorship of unknown song lyrics relies on the quantification of language, which some argue can lead to flawed conclusions (Grant 2007: 2). For example, Dr. Tim Grant argues that the underlying assumption of homogeneity within the works of a single author cannot be taken for granted (Grant 2007: 4). However, this quantification method has shown to be effective in the past. A notable recent success story comes from 2013 when linguists revealed that Robert Galbraith was in fact a pseudonym of Harry Potter author J.K. Rowling (Rothman 2013). They found the true author in the typical way—with a feature-based analysis based on a previous writing sample. A feature is a trait or pattern that can differ between text samples, and a feature space is the set of all relevant features for a single task. The idea behind this methodology is that two texts written by the same author will share more features than two texts written by different authors. Therefore, we can quantify the similarity between any two texts by computing the distance between their feature vectors (Juola 2008: 271).
The smaller the distance, the more likely it is that the texts were written by the same author.

Though the notion of features is widespread across branches of forensic linguistics, the exact feature space used depends on the task at hand. For example, two of the most helpful features when identifying J.K. Rowling were the 100 most common words and the distribution of 4-grams (strings of four characters that appear consecutively) in her adult novel *The Casual Vacancy* (Zimmer 2013). However, these features may be less indicative of a specific author in a different context. If my goal is to identify the author of a song, I have to ensure that the features I select are applicable to the domain of song lyrics.

For example, in a real forensic linguistics situation, a linguist might have to determine the author of a ransom note. In this case, the layout of the words could influence the prediction. However, it would be unhelpful for my project to include a feature relating to punctuation because the written form of song lyrics typically ignores all punctuation. This would make the feature a poor predictor of a single lyricist. Furthermore, different websites display song lyrics in different formats, and minor details such as punctuation could vary from source to source. For this reason, the physical layout of the lyrics on the website could not be a feature for author attribution because it is not a detail inherent to the lyrics themselves. Another example of an irrelevant feature is the analysis of where the line breaks occur within the body of the lyrics. Websites tend to carefully select where to put line breaks to both match up with the structure of the song and visually appeal to the reader. Together, these are some clear examples of why the
feature space is dependent on the goal of the prediction system and the nature of the texts being analyzed.

**Feature Selection**

With all this in mind, the primary linguistic goal of this project was to compile a set of features that best distinguish one musical artist from the rest. By manually inspecting a sample of artists and their lyrics, I could find some initial patterns that suggested viable features for this artist attribution task. I will present specific examples from my manual inspection that led me to some of the features I actually incorporated into my model before outlining the final list that I worked with. The first sample data set contains lyrics from the electronic pop band Owl City:

(1) Everywhere I look I see green scenic sublime / And all those oceanic vistas are so divine (Owl City, “Fuzzy Blue Lights”)

(2) With your ear to a seashell / You can hear the waves / In underwater caves (Owl City, “The Saltwater Room”)

(3) Asleep in a warm cocoon / We dream of lovely things (Owl City, “Butterfly Wings”)

(4) I'll travel the sub-zero tundra / I'll brave glaciers and frozen lakes (Owl City, “The Tip Of The Iceberg”)

We can see in (1-4) that songs performed by Owl City tend to use uncommon words such as *sublime, cocoon, and tundra* at a surprisingly high rate. We can quantify this rareness by using the inverse document frequency (IDF) to incorporate the prevalence of the word across all songs into the word distribution. If we add IDF to our feature space, the words
found in Owl City songs would be tagged as more rare than words found in the songs of other artists. Therefore, if we see lyrics from an unknown author with a similar average rareness, our algorithm would be slightly more likely to attribute authorship to Owl City than another artist. A slightly more technical description of IDF can be found at the end of this section.

Another potential feature relates to the proportion of curse words in songs by the same performing artist:

(5) Will Smith don’t gotta cuss in his raps to sell records / Well I do, so f*** him and f*** you too! (Eminem, “The Real Slim Shady”)

(6) So mow the f***** lawn, your a**** are blades of grass (Eminem, “Groundhog Day”)

For example, explicit language such as in (5) and (6) finds its way into many Eminem songs, and an unknown song that contains similar profanity should have a higher likelihood of being matched with Eminem than with Owl City. In fact, this feature could be used to distinguish entire genres from one another (see Future Work section).

The presence of interjections can also distinguish the pop genre from other genres. Looking at a sample of One Direction lyrics, for instance, it is clear that the boy band has an affinity for “oh”s and “yeah”s and “na”s:

(7) You don’t know / Oh oh / You don’t know you’re beautiful” (One Direction, “What Makes You Beautiful”)

(8) “I think it went oh, oh, oh / I think it went yeah, yeah, yeah” (One Direction, “Best Song Ever”)
If we count the number of interjectional words that appear over the course of a single song, the model can gather intuition about whether the song was performed by a boy band, which will hopefully lead it to predicting an artist similar to One Direction. Of course, there are artists in different genres that also use lots of interjections in their lyrics, but the presence of the other features in the model helps direct the classifier towards making correct predictions.

The following is a final list of the features I incorporated into my model:

**Interjection Frequency**

This was calculated by counting the number of words in the song that appear in the list ["yeah", "oh", "ohh", "ohhhh", "ah", "hey", "na", "whoa", "la"]. This list was constructed manually to cover all the words that I wanted this feature to encompass.

**Repetition**

This calculated the number of repeated lines in the song. The rationale behind having this feature is to identify artists that tend to have songs with multiple choruses, since it might be helpful for the model to determine the structure of an unknown song before matching it to an artist. This feature would also be used to recognize artists that often repeat song lines that are not part of the chorus, such as during a bridge section or a fade out ending.

**Swear count**

The model also took into account the number of curse words that appeared in the song. I downloaded a list of curse words that Google filtered out from its What Do You Love? Project and used it to find the total number of swears in a song. I thought about using a
percentage of profane words instead of a count, but I realized that this would not isolate rap songs as much as I wanted, which tend to have more words in the lyrics. Therefore, I chose to use the count under the assumption that it would be a better reflection of how willing the artist was to swear.

Average word length
I calculated the average number of characters in each word, hoping that certain artists would display a preference for words of a certain length. This was a feature that I did not expect to make a significant difference due to the minimal variation in word length, but I thought it would be worth experimenting with in case I underestimated its relevance.

Word count
As I discussed in the swear count section, some artists use more words per song than others. This may be affected by the genre of the song, but it also relates to the length of the song and the duration of instrumental solos, both of which are artist-dependent.

Inverse document frequency
This feature was part of the already-implemented code that I downloaded from scikit-learn. As I introduced above, inverse document frequency (or IDF) is a word distribution calculator that adjusts for the rareness of the word. For example, if the word “the” shows up thirty times in a song, the feature value that goes into the model for “the” is going to be lowered because the word also shows up in many other songs. However, if a word like “rainbow” shows up three times, the model will treat it as a more important feature. IDF helps train the model to focus on the relevant differences between artists’ lyrics rather than the frequency of the most common English words.
Database Construction and Preprocessing

Once I decided on a feature space for the song lyrics authorship attribution task, I began gathering the resources I needed to build the classifier model itself. Of course, I needed access to a large database of song lyrics from a large number of artists. The features I described in the previous section (and some others I will explain later) are too specific to be found in an existing database, so I had to manually construct them from accessible resources. I chose to scrape content exclusively from azlyrics.com to avoid slight differences between the lyrics from multiple websites. I chose azlyrics.com because it stored the lyrics in an easy-to-find location within the HTML code and the website is structured in such a way that iterating through all songs by a certain artist is computationally simple. However, once the lyrics were scraped, I had to work through some preprocessing steps to ensure that I was isolating the lyrics in the text files without accidentally including some of the metadata.

During the preprocessing stage, I first removed any lines completely surrounded by hard brackets [ ]. This was because some of the lyrics included indicators of the sections of the song, such as where the chorus began and which member of a band sang which verse. Because the classifier would mistake this information as lyrics of the song, I had to remove all hard-bracketed material. Similarly, I removed all songs with any featured artists. I am defining the “author” of the song to be the performing artist (more about this later in this section), so songs with multiple performing artists will interfere with the classifier’s ability to predict a single author. A more practical reason to avoid these songs is that a rap verse by a different artist in a song, for example, may have dramatically different lyric features compared to the rest of the song, which would render
all the features useless in predicting the artist. Other minor changes I made were removing any parenthetical lines at the beginning of the song (because some lyric pages used parentheses instead of hard brackets to indicate metadata) and removing all punctuation. Punctuation would be a relevant feature for authorship attribution in another domain, but because I worked with song lyrics that were typed up by various people across the Internet, I could not rely on any consistent punctuation patterns, even within the lyrics of a single artist.

Once I finished transforming all the lyrics to a usable format, I had to make the important decision about how large I wanted the database to be. It is obviously more difficult for a machine to accurately label new data if there are thousands of possible artists; however, reducing the number of artists in the database makes my results less impressive. An extreme example of this is a classifier that just learns to distinguish Eminem songs from Owl City songs—we have already seen several features that could easily quantify the stark differences between the two artists. Therefore, size was a parameter I experimented with as I constructed my learning algorithms. One helpful solution that I found was to choose a smaller number of artists to run the classifier on, but randomize which artists were fed into the model with each iteration. This not only made the classifier more accurate (because there were fewer choices for each set of lyrics in the testing set) but also ensured that the model was effective at classifying any artist in the entire set, not just a select few. I left this as an adjustable parameter so I could easily select the number of artists I wanted to include for each test.

Furthermore, it was important to prioritize artists that have a large song catalogue—i.e. force the machine to focus on predicting artists with many songs more
than artists with one or two songs. The first rule of any statistical testing is that more data is always better, and this rule applies to machine learning as well. More songs for a certain artist means more feature information and therefore more fine-tuned feature values and increased prediction accuracy. Because of this fact, I limited my lyric scraping to only pull data from artists with at least 100 songs on azlyrics.com. However, some of these artists had songs that were removed during the preprocessing stage. Therefore, I limited each artist to contribute 80 songs to the training set during each classification step. This adjusted for situations where one artist would be predicted as the artist of an unknown set of lyrics more often simply because there were more examples of that artist’s work in the training set, which improved the results.

As I hinted at previously, the most important question to consider is that of the definition of authorship. In many cases, the actual lyricist for a song is difficult to find, as the metadata only lists the performing artist. In the rare cases when a composer or lyricist is mentioned by name, there are often two or three names listed. While this would not be a problem if every song had multiple names listed, the inconsistency across the board forces me to turn instead to performing artist as my definition of author. This becomes complicated, however, because the performing artist does not always compose lyrics for his/her own songs. In fact, there are lyricists whose job is to write for multiple people.

However, it may be just as interesting to alter the goal of the project to determine the performing artist of a set of lyrics, ignoring who was actually responsible for selecting the exact words. It is often the case that a band or singer will choose to perform songs similar in sentiment and style even if the lyrics were written by other people. Therefore, my definition of “author” is the performing artist, as I believe the results of the
classifier produced valuable and interesting information. However, this definition may be fine-tuned in future work once it can be determined whether enough lyricist data exists to train and test a computational system.

**Supervised Classification**

After this was all decided, I began focusing on the computational portion of the project: constructing my model. This project would not be possible without the extensive previous work in the field of machine learning, which is the idea that computers can learn about a data set and make accurate predictions automatically with minimal explicit programming. This is obviously a huge task, so it is broken down into more specific types, each designed to solve a different problem. For my purposes, I focused on supervised classification, which involves predicting information about each element in an unknown data set after having trained on a data set with known labels. There are several algorithms that help with this, such as decision trees, Naïve Bayes, and k-nearest neighbors, all of which generate predictions in different ways. Although the algorithms are different, they all rely on the features of the training set to make their predictions. I have already discussed the process by which I selected relevant features, and the next step was to select which algorithm was the most appropriate for the lyric authorship attribution task. I will now briefly explain the algorithms that I tested with my classifier and describe how they made their predictions and emphasize the differences between their methods.

**Naïve Bayes**
The Naïve Bayes model relies on a strong assumption that each feature value is independent of the rest, which may or may not always be the case. By assuming independence, however, we can sum the probabilities of each feature value being generated by a data point with a specific output label. This allows us to find the classification most likely to produce the complete feature vector in question. At first glance, this process seems backwards—we are hoping to find the probability of each output label given the feature vector (so that we can choose the highest as our prediction) yet we are calculating the probability of the feature vector being generated given an output label. However, we can achieve the same result by selecting the label that has the highest probability of producing the feature vector. This is the fundamental strategy behind the Naïve Bayes classifier, and it may be appropriate for a song lyrics domain because it doesn’t require a complicated training process (Kotsiantis 2007: 257). Because our training data set for each artist was relatively small, a model that could still effectively function was essential.

Decision Tree

An alternate model is a decision tree, which assigns classifications to new data by learning a set of rules that most effectively divides the data by label. When organized into a tree structure, we can classify an unknown point by passing it through the tree and sorting by its feature values (Kotsiantis 2007: 251-252). The primary benefit of decision trees is their simplicity, though they have been shown to be effective at supervised classification in certain situations.

Support Vector Machines
Support Vector Machines work by separating the data into distinct classes with a linear hyperplane. This method works best if the data is linearly separable, because then the hyperplane can cleanly determine which group new data points belong to (Kotsiantis 2007: 260). However, it is difficult to gather data that is completely separable, so I did not expect SVMs to be particularly effective on the corpus of song lyrics because many artists had overlapping feature vectors.

**K-Nearest Neighbors**

The K-Nearest Neighbors classification technique relies on the assumption that data points with similar feature vectors will be classified the same way. The algorithm works by locating the k-closest vectors to the test vector and labeling the test vector with the most frequent classification of the nearby vectors (Kotsiantis 2007: 259-260). When I ran the algorithm on my data set, I did not change the default value of $k = 5$.

**Random Forest**

A random forest is simply a combination of multiple decision trees that outputs the most common classification of each tree. Therefore, the algorithm itself is not more complicated than that of the decision tree, but the results are likely to be more accurate because there are more data points being accounted for in the overall classification.

Clearly there were a variety of options for which classifier I should use for this task. Therefore, I ran experiments with all the classifiers to elicit quantitative evidence regarding which one performed the best within the domain of song lyrics. To make my job easier, there were pre-existing implementations of all these supervised classification algorithms. This means that I could focus my time and energy on narrowing down the
best feature space and ensuring that the preprocessing was comprehensive and spend less time writing the code to actually perform the automatic classification. With a feature space already determined and the preprocessing complete, I was able to begin testing the classifiers and gathering data.

Results

All of the following results were run selecting from a set of 413 artists (after the preprocessing removed 159 of them from the corpus). For each test, the program performed the cross-validation classification on a set of twenty randomly selected artists. I tested the model with different size datasets, and I found that twenty was an appropriate balance between having enough candidate artists to make the results interesting and getting the accuracy high enough that the scores were comprehensible. However, I did find that the ratio of the model’s success compared with chance remained consistent throughout the tests, showing that the model does work as expected with differently sized datasets.

This first table shows how each classifier performed on a set of 20 randomly chosen artists first using the following six features – number of interjections, number of repeated lines, number of swears, average word length, word count, and word distribution (including IDF)—and then using just the word distribution (without IDF). I performed these two experiments because I expected the word distribution to be the most important and beneficial feature. If I had tested each combination of classifier, subset of features, and number of artists, there would have been hundreds of experiments to run, making a clean analysis and discussion virtually impossible. Therefore, the goal of this initial test
was to determine which of the five classifiers worked best for this task so I could use it for the remainder of the results section. Each score (with standard deviation in parentheses) is the average of ten trials, each with a randomly selected group of twenty artists:

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Score (std), all features</th>
<th>Score (std), just word distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>0.182 (0.024)**</td>
<td>0.518 (0.051)</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>0.142 (0.019)</td>
<td>0.147 (0.043)</td>
</tr>
<tr>
<td>K-Nearest Neighbors</td>
<td>0.138 (0.024)</td>
<td>0.206 (0.047)</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.279 (0.048)</td>
<td>0.295 (0.053)</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.270 (0.025)</td>
<td>0.256 (0.039)</td>
</tr>
</tbody>
</table>

**with all features except IDF, the Naïve Bayes classifier got a score of 0.539 (0.041)**

The scores in the above table were calculated by taking the weighted average (harmonic mean) of the precision and the recall of the classifications. To better understand these statistics, let us focus on all lyrics in the test set that were attributed to Owl City. The precision is the number of songs in this set that were correctly classified as Owl City songs divided by the total number of songs in the set. The recall is the number of songs in this set that were correctly classified as Owl City songs divided by the total number of Owl City songs (regardless of whether our model correctly labeled them or not). Therefore, the higher the score, the better our model’s results are.

One immediate conclusion from these results is that the Naïve Bayes classifier performed dramatically better than the other four classifiers when only using the word distribution as a feature. With twenty possible artists for each instance in the test data, we would expect a program that guesses randomly to be correct five percent of the time (1/20), which would be a score of 0.05. Therefore, we can conclude that the Naïve Bayes classifier using the word distribution feature, which classified over half of the test data
correctly, performed far above a random classifier would have done. It is concerning that we see that when I added in the features that I built myself, the score decreased to 0.182. However, with further testing, I determined that running the model on all features except IDF led to a score of 0.539, showing that the IDF feature and the Naïve Bayes classifier did not combine well for this task. Further inspection led me to realize that the classifier had predicted a single artist far more than any other artists (although not exclusively). At this point, I do not have a satisfying explanation for why this feature negatively affects the Naïve Bayes classifier so significantly, but I hope that an extension on the model will consider this problem further.

Another interesting comment is that in all cases except Random Forest, adding the additional features decreased the overall score, although most did not decrease significantly. However, the minimum score, courtesy of the K-Nearest Neighbors classifier, was still over double the expected score for a random classifier, showing that the features that I selected did contribute to the success of the model. I interpreted the increase in the Random Forest classifier’s score to mean that it was the best model for this domain, as the features I wanted to study in depth combined to correctly predict an average of 27% of the artists in the test set. Therefore, I decided to use the Random Forest classifier to determine which of the features (besides the word distribution) is the most effective predictor of a song’s artist.
As was expected based on the results from Table 1, the word distribution feature was significantly better than any of the others, whether it combined with IDF or not. The reason that I was unable to isolate the inverse document frequency is that it can only be applied in conjunction with the word distribution. This is because, as a reminder of its function, the IDF discounts the word frequency counts depending on the total number of songs that the word appear in. It also appears that the only features that reliably perform better than chance are repetition, average word length, and word count. This is somewhat disappointing, as I had hoped that the features I chose would show marked improvements over a baseline chance model. However, it is still informative to learn that interjection counts and profanity frequency may not be as indicative of the performing artist than I had thought.

After testing each of the features independently, I wanted to run a final experiment to determine whether combining any of them would elicit better results than their individual performances (sticking with the Random Forest classifier and excluding the word distribution and IDF features). Because there were 31 possible combinations of features, I strategically selected the few that I wanted to test. I prioritized features that

<table>
<thead>
<tr>
<th>Feature</th>
<th>Score (std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+interjections</td>
<td>0.032 (0.012)</td>
</tr>
<tr>
<td>+repetition</td>
<td>0.064 (0.011)</td>
</tr>
<tr>
<td>+swears</td>
<td>0.053 (0.015)</td>
</tr>
<tr>
<td>+avg word length</td>
<td>0.069 (0.010)</td>
</tr>
<tr>
<td>+word count</td>
<td>0.098 (0.008)</td>
</tr>
<tr>
<td>+word distribution</td>
<td>0.286 (0.057)</td>
</tr>
<tr>
<td>+word distribution/IDF</td>
<td>0.301 (0.048)</td>
</tr>
</tbody>
</table>
had done well in the individual tests, although I also wanted to see how a combination of good and poor features would perform.

<table>
<thead>
<tr>
<th>Features</th>
<th>Score (std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+repetition +avg word length</td>
<td>0.087 (0.008)</td>
</tr>
<tr>
<td>+repetition +avg word length +word count</td>
<td>0.150 (0.019)</td>
</tr>
<tr>
<td>+repetition +swears +avg word length +word count</td>
<td>0.176 (0.030)</td>
</tr>
<tr>
<td>+interjections +repetition +swears +avg word length +word count</td>
<td>0.205 (0.027)</td>
</tr>
<tr>
<td>+avg word length +word count</td>
<td>0.123 (0.023)</td>
</tr>
<tr>
<td>+interjections +word count</td>
<td>0.115 (0.016)</td>
</tr>
<tr>
<td>+interjections +swears</td>
<td>0.083 (0.018)</td>
</tr>
</tbody>
</table>

When I combined multiple features, the scores increased. We would hope this to be the case, for it would be counterintuitive for the model to make worse predictions given more information about each artist. However, not only do we notice that the scores increased, we also notice that the amount that the scores increased when a new feature was added is roughly proportional to the impact that the original feature had when applied to the data set on its own. This further supports the idea that each of these features improved the quality of the results, though at differing levels of success. For example, just using the interjections feature led to a score of 0.032 and just using the swears feature led to a score of 0.053. Using both features got a score of 0.083, which is almost equivalent to the sum of their individual scores. This pattern occurred throughout all of Table 3, where adding new features only improved the quality of the results.

The final test I did ensured that the model was adept at predicting the artist even when there were many more options. I ran every feature on a randomly chosen set of 50 artists ten times using a Random Forest classifier and averaged the results. The model
received a score of 0.181 with a standard deviation of 0.027. Although this appears low, remember that it should be compared to the average performance of a random classifier, which we expect to get a score of 0.02 (1/50). With this in mind, we see that the resulting score still demonstrates the ability of the features I selected to predict the performing artist of a song given its lyrics.

**Future Work**

There are additional features that might be useful to incorporate into the model in the future. There were various reasons that I did not include them here, ranging from the amount of information that I scraped from the Internet to the difficulty of incorporating third party tools in a reliable and useful way. However, I will briefly describe them here in hopes that any expansion on this model can integrate them in some way.

First of all, the length of song titles could be a distinctive feature of certain artists:

(10) “Our Lawyer Made Us Change The Name Of This Song So We Wouldn't Get Sued” (Fall Out Boy, *From Under the Cork Tree*)

(11) “I’ve Got A Dark Alley And A Bad Idea That Says You Should Shut Your Mouth” (Fall Out Boy, *From Under the Cork Tree*)

(12) “The Only Difference Between Martyrdom and Suicide is Press Coverage” (Panic! At the Disco, *A Fever You Can’t Sweat Out*)

(13) “Lying Is The Most Fun a Girl Can Have Without Taking Her Clothes Off” (Panic! At the Disco, *A Fever You Can’t Sweat Out*)

Bands in the pop-punk genre such as Fall Out Boy and Panic! At the Disco are notorious for having obnoxiously long song names (see 10-13). Therefore, it may be
beneficial for the classification model to learn that an unknown lyric set with a fifteen-word title has a higher probability of being written by Fall Out Boy than Owl City. This length metric can be generalized not only to include album names but also to incorporate the number of total words in each song (something I did include in my classifier), as artists from different genres tend to produce songs of different lengths. For example, rap artists tend to fit more words in each song due to the fact that rapping is based on a high number of words per minute. On the other hand, the distinctive components of dubstep music are the instrumental sections, so songs tend to have fewer words.

In general, artists can sometimes be identified by the recurring concepts they incorporate into their songs. For example, the Owl City quotes at the beginning of the paper demonstrate different common themes such as water and dreams. Furthermore, many Eminem songs contain powerful imagery and disturbingly casual discussions of blood and violence. In order to use a semantic feature, however, we would need a way to automatically determine which words belong to similar categories. Luckily, WordNet—a large and free-to-use lexical tree-shaped database that links related words—may be useful for this task. A potential quantification of thematic similarity is to count the number of hierarchical levels between the words in the lyrics and the theme word in WordNet’s tree structure, but the feasibility and reliability of this method will have to be explored further.

Predicting more than just the artist of a song may actually improve the accuracy of the author attribution system. Previous work has delved into predicting the genre, time period, and quality of popular songs based on the lyrics (Fell 2014: 629). This has been done using a statistical model similar to what I propose here, and its success gives me hope regarding the success of my authorship attribution model. Some key features used in
Fell & Sporleder 2014 were the rhyme structure, prevalence of echoisms (repeated letters or words), and frequency of curse words. Not only would these features be appropriate for this task’s feature space, but the researchers also found them to be accurate predictors of a song’s genre. For example, they discovered unsurprisingly that rap songs tend to have a higher frequency of curse words (Fell 2014: 626). Other genres also displayed distinguishing features that the classifier was able to recognize in a set of unknown lyrics. One methodology to consider adding to my classifier is to first select the genre of the lyrics and then to only attribute them to artists within that genre. This could be a good way to narrow down the pool of potential authors from the start, and I expect that this two-tiered method would actually improve the prediction quality.

Conclusion

While there has been previous work related to artist attribution for song lyrics (Mara 2014: 1-5), I hoped to expand the scope of such classifiers and analyze the effects of different strategies. My results were unremarkable yet informative, as I found that a Random Forest classification model trained on a combination of carefully selected features was able to identify the performing artist of a given set of lyrics at a higher-than-chance rate. However, the actual accuracy (hovering around the 25% mark with a set of 20 artists) has room for improvement. I have outlined potential expansions for the model and explained why I expect them to improve the quality of the results. I would also like to look more closely into the effects of database size and feature weights, as both could also help increase the predictive accuracy of the model. Overall, bringing authorship attribution within the domain of song lyrics seems like a manageable step away from
previous work and I am optimistic about this model’s success after the aforementioned
tweaks are made to further improve its performance.
References


