The Ghost-Writing Phenomenon:   
An Application of Authorship Detection Techniques to Hip Hop Music

**Background/Approach:**

As the developments and computational techniques in the field of digital humanities, computational linguistics, and/or natural language processing continue to grow at lightning fast speed, it makes it difficult for the applications of these methods to keep up. In recent years, a large amount of texts have become digitized, offering a new opportunity to use computers to analyze these corpora. Much of the work in this space has been focused on analyzing popular literature because it is much more practical for that academic discipline. However, many of the findings and tools used in this field can be adapted to use for other bodies of texts.

For example, authorship detection has received a lot of attention specifically in reference to novels and historical documents of importance. However, I am personally interested in music, specifically hip hop and rap music. In addition to true rappers and artists, one profession that has always thrived in hip hop music is that of the ghostwriter. A ghostwriter in rap tries their best to write lyrics to a song which will ultimately be recited by another artist. An unwritten rule in ghostwriting is that a ghost writer should never reveal themselves. The practice of discretion is valued just as much if not more than the ability to write good music. Only on rare occasions has this golden rule been broken, oftentimes resulting in the ostracizing of the ghost writer.

In my final project I attempted to adapt a simple method used for authorship detection in novels to the rap lyric data I collected. Contrary to the aforementioned golden rule, there are many cases where the ghost writer of a song is commonly known, as a result of speculation or gossip or leaked information. My approach was to leverage these known cases of ghost writing to tune my algorithm to a certain point of accuracy and then apply the algorithm to disputed cases of ghostwriting (meaning the artist has spoken out and denied that someone else had written their music). To do this, I needed to collect multiple song lyrics for each artist involved in either the known cases or the disputed case, as seen below:

Disputed Cases**:**

1. **Credited Author:** Nas; **Disputed Author(s):** Jay Electronica & Stic Man; **Album:** Untitled
2. ***Credited Author:***  *Lil Wayne;* ***Disputed Author(s):*** *Gillie Da Kid ;* ***Album:*** *The Carter I\**

Common Knowledge Cases:

1. **Real Author**: Nas; **Artist:** Will Smith; **Song**: Gettin Jiggy With It
2. **Real Author:** Jay-Z; **Artist:** Dr. Dre; **Song**: Still D.R.E.
3. **Real Author:** Pharoahe Monch; **Artist:** P. Diddy ; **Song**: The Future
4. **Real Author:** Skillz VA; **Artist:** P. Diddy; **Song**: Bad Boy For Life

**\****Unable to investigate this case because RapGenius did not have enough lyrics for Gillie Da Kid*

**Data:**

My rap lyric collection was the result of manual data parsing from the popular lyric website RapGenius.com. This website offers the lyric texts of songs from multiple artists and multiple genres. For each artist of entry, I simply created a directory (or multiple directories if they are involved in multiple cases) and copy/pasted their lyrics into a plaintext file. I also did this for each specific disputed song. I had to do this manually because there are three specific scenarios where the data needs to be modified from the RapGenius format: 1) If there are any featured artists in a song or then those lyrics need to be erased because I do not want them to be accredited to the artist of interest during feature extraction 2) RapGenius does not only contain the lyrics that are recited in the song, it also provides texts such as ad-libs or an intro/outro to a song that may just be the artist talking, not actually rapping 3) RapGenius supplies meta-text on the page such as *[Verse 1]* or *[Hook]* to signal to the reader exactly which part of the song the subsequent text belongs to. Because the annotation of these scenarios is not standardized across all songs I could not simply scrape the website and ignore certain strings of text.

This term we have seen how drastically writing styles can change over time, so to avoid collecting songs that are not representative of the artist’s writing style when the ghost written or disputed ghost written song came out, in each case above, I collected song lyrics from the album released most closely to the song of interest. Oftentimes each artist had an album come out either a few months prior to or after the disputed song, a time frame unlikely to allow for drastic stylistic change. However, this introduces a data limitation factor because instead of collecting the entire catalog of an artist, I am limited to songs released during a certain window of time. However, my hope was that the accuracy of my algorithm would not be correlated with the volume of my corpus.

**Methodology:**

Throughout the term I have seen the use of probability and statistics to determine authorship of a disputed text. These methods have been applied to the famous *Federalist Papers* dispute between James Madison and Alexander Hamilton, as well as novelistic data comparing works J.K. Rowling and P.D. James. I used this data from our class as control data to gauge the accuracy of my algorithm.

My goal was to choose an algorithm that was as simple to implement as something like Bayes’ rule so that it would be accessible to non-mathematical analysts of text, but sophisticated enough to let me capture more than a single feature as we saw in our Bayes’ models. I decided to choose a K-Nearest Neighbors algorithm. More specifically I chose to do a majority voting KNN algorithm which works as follows:

* Input: A set of text documents (consisting of work from two different authors), the comparison document, and a set of features to be analyzed
* Algorithm:
  + For each features in the set of features to be analyzed do the following:
    - Create a vocabulary representative of the selected feature across all documents (i.e. a vocabulary of every character trigram that appears in the set of all documents)
    - For each document in the set create a high dimensional point representative of the rate of each item in the vocabulary presence in that document
    - Perform KNN on the comparison document’s high dimensional point against every other document in the set
    - Determine if the K Nearest Neighbors are from author 1 or author 2. Tally one vote for the appropriate author
  + Determine which author got a majority of the votes across all features that were analyzed and predict that this is the author who is responsible for writing the text

**Python Files:**

*authorship.py:* This is the main file for my project. This program takes three parameters: 1) The path of the directory containing text files written by author one 2) The path of the directory containing text files written by author two 3) The name of the disputed text file OR the name of the directory containing a set of disputed text files to be analyzed. For example: *python authorship.py Hamilton Madison hamilton28.txt* will predict whether Madison or Hamilton is the author of hamilton28.txt based on the KNN algorithm described above.

*validate\_authorship.py:* This file takes three parameters: 1) The path of the directory containing text files written by author one 2) The path of the directory containing text files written by author two, which will each be analyzed independently and have its author predicted. It runs a starkly similar algorithm to *authorship.py* except it holds out one document from the second author on each iteration and uses that document as the “disputed file” for author prediction. I used this file to gauge the accuracy of the algorithm. For example: *python validate\_authorship.py* *Hamilton Madison* should predict the author of every document in the Madison directory to be James Madison.

*file\_parser.py:* This module supports 1) parsing the file to extract plain text lyrics (regular expressions to get leave out unwanted characters) 2) feature extraction. The algorithm currently uses the following features: word vocabulary, character four-grams, word bigrams, song length (characters), average word length, line count, and average line length.

*knn.py:* This module supports the KNN algorithm. When given a set of points, a comparison point, and a value for *k*, it will return the *k* closest neighbors to the comparison point from the first set.

**Results:**

My results from this analysis are inconclusive to say the least. Because we have seen analysis of the *Federalist Papers* and the Rowling vs. James comparison in class, I used these cases as control cases to gauge whether my algorithm seems appropriate or not before branching to analyzing music. It is rather strong in predicting the James vs. Rowling case, but seems to almost always favor Hamilton when comparing to Madison. Results of these comparisons can be seen in the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hamilton Vs. Madison (validate\_authorship.py)** | | | | |
| **Text** | **Predicted Author** | | **True Author** | |
| madison10.txt | Madison | | Madison | |
| madison14.txt | Hamilton | | Madison | |
| madison40.txt | Hamilton | | Madison | |
| madison41.txt | Hamilton | | Madison | |
| madison42.txt | Hamilton | | Madison | |
| hamilton15.txt | Hamilton | | Hamilton | |
| hamilton16.txt | Hamilton | | Hamilton | |
| hamilton25.txt | Hamilton | | Hamilton | |
| hamilton26.txt | Hamilton | | Hamilton | |
| hamilton27.txt | Hamilton | | Hamilton | |
| **Text** | **Predicted Author** | | **True Author** | |
| TheCasualVacancy00 | Rowling | | Rowling | |
| TheCasualVacancy01 | Rowling | | Rowling | |
| TheCasualVacancy02 | Rowling | | Rowling | |
| TheCasualVacancy03 | Rowling | | Rowling | |
| TheCasualVacancy05 | James | | Rowling | |
| TheCasualVacancy06 | Rowling | | Rowling | |
| TheCasualVacancy07 | Rowling | | Rowling | |
| TheCasualVacancy08 | Rowling | | Rowling | |
| TheCasualVacancy09 | Rowling | | Rowling | |
| TheCasualVacancy10 | Rowling | | Rowling | |
| TheCasualVacancy11 | Rowling | | Rowling | |
| TheCasualVacancy12 | Rowling | | Rowling | |
| ThePrivatePatient00 | James | | James | |
| ThePrivatePatient01 | James | | James | |
| ThePrivatePatient02 | James | | James | |
| ThePrivatePatient03 | James | | James | |
| ThePrivatePatient04 | Rowling | | James | |
| ThePrivatePatient05 | James | | James | |
| ThePrivatePatient06 | James | | James | |
| ThePrivatePatient07 | James | | James | |
| ThePrivatePatient09 | James | | James | |
| ThePrivatePatient10 | James | | James | |
| ThePrivatePatient11 | James | | James | |
| ThePrivatePatient12 | Rowling | | James | |
| **Known Cases of Hip Hop Ghost Writing (authorship.py)** | | | | |
| **Song** | **Credited Writer** | **Predicted Writer** | | **True Writer** |
| Getting Jiggy With It | Will Smith | Nas | | Nas |
| Still D.R.E. | Dr. Dre | Jay-Z | | Jay-Z |
| The Future | P. Diddy | P. Diddy | | Pharaoahe Monch |
| Bad Boy For Life | P. Diddy | Skillz VA | | Skillz VA |
| **Disputed Cases of Hip Hop Ghost Writing (authorship.py)** | | | | |
| **Song** | **Credited Writer** | **Predicted Writer** | | **True Writer** |
| We’re Not Alone | Nas | Jay Electronica and/or Stic.Man (5/7) | | Disputed |
| Untitled | Nas | Jay Electronica and/or Stic.Man (6/7) | | Disputed |
| Queens Get the Money | Nas | Nas (4/7) | | Disputed |
| Sly Fox | Nas | Nas (6/7) | | Disputed |

**Conclusion:**

Because the results of my control tests (the Hamilton vs. Madison, Rowling vs. James, and known hip hop cases) are not indicative of a consistently accurate algorithm it is difficult for me to make sense of the results when I apply it to novel data. However, because I use seven different features in my KNN algorithm, I do have some sort of confidence factor. Based off of the votes seen in the predicted writer column above, we can see that for songs like *We’re Not Alone, Untitled,* and *Sly Fox* the algorithm was rather confident in its selection of the author because five or six of the seven features agreed upon the author. In the case of *Queens Get the Money* we see that it is a much closer decision and therefore much less confident.

I think this question of authorship detection in rap lyrics poses a lot of challenges. This detection task is different than that of the *Federalist Papers* and ghost writing in novels, because part of the job of ghost writing in rap is to try to imitate the lyrical pattern of the person who has hired you. Conversely, in the case of the *Federalist Papers*, Hamilton and Madison were not trying to imitate each other’s writing style by any means. Furthermore, in rap in general, artists are praised for their ability to write different “flows” and to switch up their style from song to song. A rapper who has a repetitive or too overtly stylistically familiar flow or lyrical pattern would be frowned upon or seen as an amateur. So part of a rap artist’s job is to disguise his or her own style in a way that it does not become too repetitive. The fact that ghost writers can imitate another rapper clearly means that rappers aren’t perfect at disguising their style, but nonetheless it presented an unexpected challenge for me and I think it makes it all the more difficult to extract consistent features from an artist’s catalog.

Overall I think my intuition and approach to the problem was a reasonable idea, but again the results were inconclusive. I do not think a model as simple as Bayes’ rule or a modified KNN is sophisticated enough to learn the characteristics of a rapper to distinguish him or her from other rappers.

**Sources:**

"10 Ghostwritten Hip-hop Tracks – and the Surprising Ghostwriters behind Them – FACT Magazine: Music News, New Music." *FACT Magazine Music News New Music RSS*. FACT, 20 Jan. 2014. Web. 13 Mar. 2014.

Hacking, Ian. "Bayes' Rule." *An Introduction to Probability and Inductive Logic*. Cambridge, U.K.: Cambridge UP, 2001. Print.

Halvani, O., Steinebach, M., Zimmerman, R. “Authorship Verification via *k-*Nearest Neighbor Estimation.” *Notebook for Pan at CLEF 2013.*

Mosteller, Frederick. “A Statistical Study of the Writing Styles of the Authors of ‘The Federalist’ Papers.”Proceedings of the American Philosophical Society 131, no. 2 (June 1, 1987): 132–140.

*Rap Genius*. N.p., n.d. Web. 13 Mar. 2014.