Machine Learning the Authors of the Federalist Papers

https://github.com/JSunde/FederalistPapers

|  |  |
| --- | --- |
| Peter Giseburt | Jakob Sunde |
| [petergg@uw.edu](mailto:petergg@uw.edu) | jsunde@uw.edu |

**Abstract**

The Federalist Papers were originally printed under a pseudonym, Publius. Since then it has been discovered that the papers were written by Hamilton, Madison, and John Jay. However, some of the papers are of disputed authorship. To predict the authors of these papers, we will perform feature selection words and then run the algorithms: kMeans, k-NN, and Naïve Bayes to determine the authorship of the disputed papers.

**1 Project description**

We examined the text of the Federalist Papers provided by Project Gutenberg. Then, using the known authors we converted the text into features and compared the features of known authors against the papers that are unknown or in dispute to determine the author.

Individual writers have individual styles, which can be represented in terms of how frequently certain words or phrases are used. From some research on the subject, certain “function words” are more indicative of a specific author, while other words, like nouns, can be highly influenced by subject matter and thus are not a good indicator of a specific writer [1].

Despite the main authors being Hamilton or Madison, John Jay is also credited with a few of the Papers. Therefore, we cannot use a simple binary classifier, as there are three potential authors. We chose to use kMeans, k-NN, and Naïve Bayes classifiers as they allow for multiple classifications.

**2 What we’ve done so far**

First, we gathered our data. We downloaded a .txt file of all the Federalist Papers from Project Gutenberg that contained all the papers, and wrote a parser that went through that file to collect the counts of the words used in each paper, which we converted into a frequency by the number of words in that paper. From knowing which authors wrote each paper, and the frequencies of words in each paper, we create an average word frequency vector for each author, as well as a global word frequency vector for all papers. We then calculated the distance from this overall average word frequency to each of the respective author’s word frequencies, for each word. We summed this distance for all the authors to find the words that had a high variation from the mean frequency, which indicates a high difference from each other. Surprisingly, these words were mostly “function words,” despite us not filtering for specific words at all. With these N (with N being the number of words that we use in our feature vector to classify texts) words, which we tested different numbers of N for, we had the features we could use to predict authors.

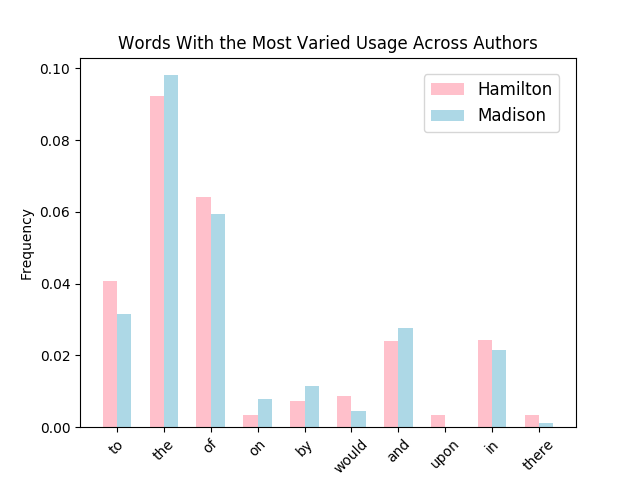
We then used k-NN to predict authors, both on the test data (unknown author papers) and the training data (known authors). We calculated the sample for each author by simply calculating the average word frequency for each of the chosen feature words over all of their known papers. We then tested the known papers against this sample for different N values, finding a large amount of error starting at N = 1, which sharply dropped until N = 5 and flattened out. For our test data, we found that it, after a shaky start, predicted Madison for 10 out of 12 papers from N=5 onwards. In Glenn Fung’s paper, he talks about how his separating plane (with only 3 variables: to, upon, and would) predicated that all 12 disputed papers were Madison’s, and other work on the subject agrees with him [1]. However, he used a separating plane, which would incorrectly classify John Jay’s papers as either Hamilton or Madison and thus Jay’s papers were ignored. All three of “to, upon, and would” appeared in the list of words that we highly valued.

Hamilton and Madison collaborated on some papers together, and as such should have an overlapping writing style for these papers, which may skew, the word frequency we expect from each of the authors. Upon removing the papers that both Hamilton and Madison collaborated on, we received similar, slightly mutated data. It had a slight increase in errors for the training data, however for high N’s the model predicated 11 out of 12 to be Madison, which is close to the generally accepted result of all 12 being Madison’s.

**3 Feature Selection**

TODO: Talk about the intuition behind why the words that have the most varied usage between authors would be good indicators of an author’s style

To featurize the data, we calculated the word frequencies of the words that had the greatest disparity of usage between the authors. This provides a way to represent the style of an author, by how frequently they use certain words. Nouns and other subject specific words are unlikely to appear in this set of words with the most varied usage. All the authors are writing on similar topics, so it is unlikely that one would use a specific noun much more often than another author. This further solidifies the validity of this featurization of the data.



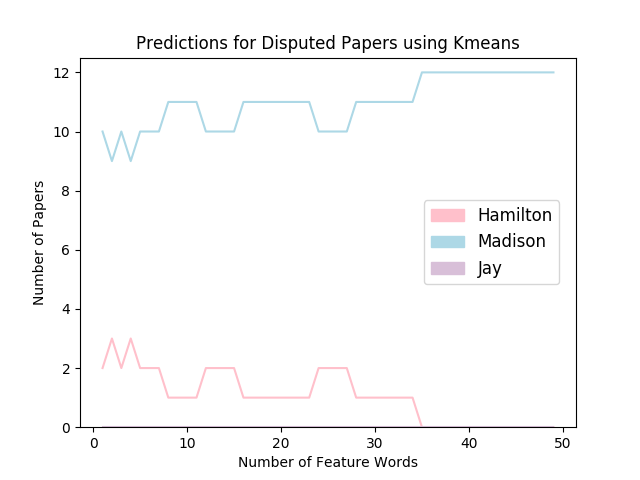
**4 k-Means**

TODO: Maybe describe the intuition behind why kMeans would work well

TODO: Describe algorithm

First we compute an average Madison “paper” and an average Hamilton “paper”. We do this by computing the word frequencies of all of the feature words as described in Section 3. Then we look at each of the disputed papers and compute the Euclidean distance between the disputed paper the two average papers. The disputed paper is labeled with the author of the average paper with the lowest distance.

We ran this implementation of k-Means varying the number of feature words and recorded the training error and author predictions. Training error will be discussed in Section 7. k-Means always predicts that a high number of the disputed papers were written by Madison, and with greater than 35 feature words, k-Means predicts that all 12 of the disputed papers were written by Madison.

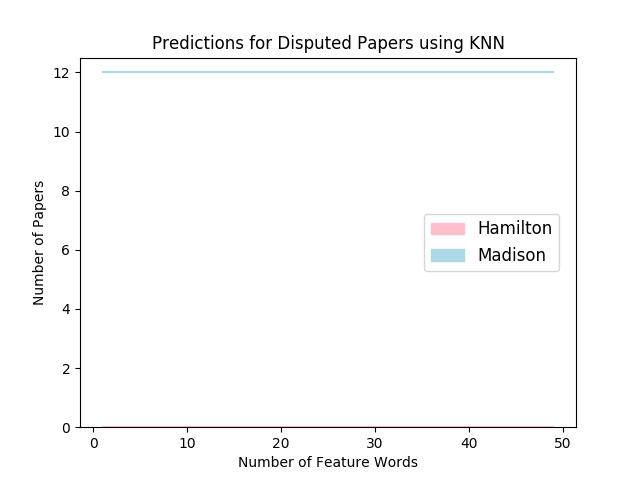


**5 k-NN**

TODO: Describe algorithm

TODO: Describe implementation

It’s surprising that k-NN is able to predict that Madison wrote all 12 papers with only one feature word. Simply looking at the frequency of the word “to”, k-NN comes to the same conclusion that previous research as agreed upon.



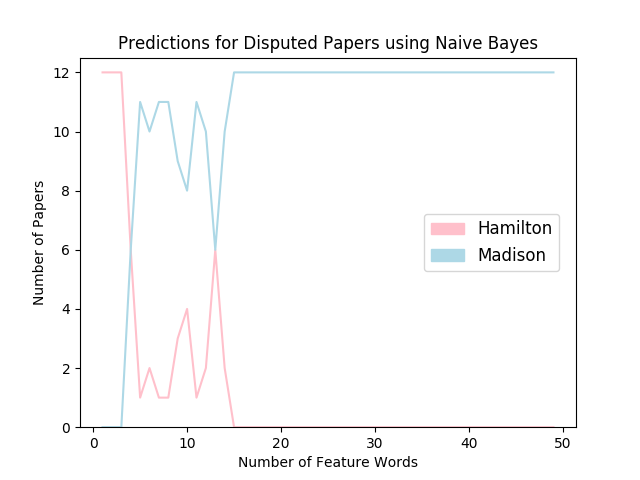
TODO: Insert Training Error Plot

**6 Naïve Bayes**

TODO: Describe algorithm

TODO: Describe implementation

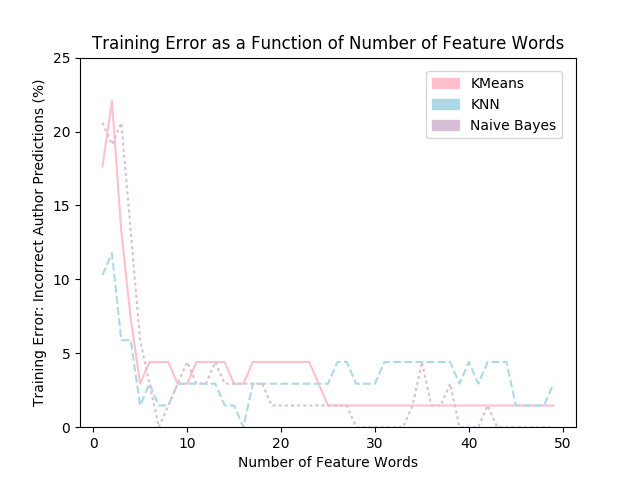
Where is the paper being examined, is the probability of any paper in the set being written by which is given by , and is the probability of a given word in existing in a paper written by .



TODO: Insert Training Error Plot

**7 Discussion of Results**

TODO: Add plots side by side and talk about the relative performance of each



TODO: Talk about the validity of the algorithms and their training error, why we think one had lower error than others, and which we think might perform best on papers that don’t have to do with the topic

**7 Discussion of Jay**

TODO: Talk about Jay’s irrelevance since we ran it with his papers included, but he was never predicted as the author in any of the algorithms

**8 Conclusion**

**9 Related Work**

Fung and Bosch’s papers used support vector machine feature selection to create separating hyperplanes of 3 dimensions. These hyperplanes correctly classify all of the training data, and predict that all 12 of the disputed papers were written by Madison. Our findings support those of these research papers, while our Naïve Bayes model might be more easily applied to papers outside of the Federalist Paper dataset, as it outputs a probability of a paper being written by a certain author, whereas the hyperplanes simply classify a paper as being written Hamilton or Madison.

**References**

[1] Fung, Glen (2003). The Disputed Federalist Papers: Svm Feature Selection via Concave Minimization. In: *Proceedings of the 2003 conference on Diversity in computing*, pp. 42–46. ACM Press, New York.

[2] Bosch, R. A. and Smith, J. A. (1998). Separating Hyperplanes and the Authorship of the Disputed Federalist Papers. In: *American Mathematical Monthly*, pp. 105, 7 (August-September), 601–608.