# Author Identification of Written Text

Ari Brown, Jen Eisenberg, Jordan VanderZwaag

April 21, 2014

## 1 Introduction

Author identification is the process of choosing the correct author of a given text based on various feature inputs for a set candidate list of authors. On the surface level, it can be trivial to identify the author of a text due to salient entities such as the identification of personal relationships, specific subjects, and maybe even tone of voice. However, it is a much harder problem to discern why one author's "voice" is more prevalent or unique than another's when writing about the same topic or across creative and academic pieces.

Thus the task at hand requires identifying many stylist writing features within a text and comparing those percentages across several written pieces by the same author to identify what features essentially make this author this specific author. The problem itself is interesting on the personal level, linguistically analyzing why one author writes in a certain style. With the data extracted in this program, one can access questions such as: what are the motivations for using words with similar etymologies? Are one's sentence structures too deeply nested? Should one vary his or her conjunction use more? Certainly, the problem can approach broader fields within realms of crime detection and anonymous revelations. However, this project attempts to solve the problem of author identification at it's core with student texts.

## 2 Related work

Author identification techniques have been applied to solve large problems within the realm of forensics, missing literary classifications, authorship attribution and text mining. In a similar study conducted by Efstathios Stamatatos entitled "Ensemble-based Author Identification Using Character N-grams", the researchers maintained usage of X different features to classify the author's "stylometry", which is a term they deemed as a combination of such features for representing the author's work. While we both employed the sentence structure feature, Stamatatos focused on n-gram frequencies as opposed to word frequencies. They chose to implement the same Support Vector Machine algorithm. However, in our implementation we had fewer pieces of text to test but we utilized more features as opposed to solely focusing on n-gram frequency. The work by Stamatatos claims that the sub-word units which characterize n-grams can reveal more about an author's lexical, semantic and stylistic choices than a comparison across various features.

## 3 Data and Methods

Data was collected from papers written throughout the college careers of the project team and their friends, for a total of 8 authors and 85 separate papers used to train the model. These papers included both creative works and more formal essays, with some authors contributing papers all written in the same style and others that contributed some of both. All extra information included in the papers, such as titles, section headings, and citations, were removed to focus solely on the writing style of the author.

To evaluate the papers, the team decided on a number of features which were thought to be unique

between authors and consistent between the author's works. In total, 14 unique characteristics were chosen and algorithms were written to measure these values for each author. These characteristics were the the average length of each sentence, the average length of each word, the rate of commas, semicolons, colons, quotation marks, and conjunctions per sentence, the average percentile position of each subject and verb in each sentence, the average max depth of a parse tree of a sentence, the average number of nodes in a parse tree of each sentence, the average number of words used which were not found in an English dictionary, the percentage of sentences that used active voice instead of passive voice, and the percentage of words of various etymologies.

To find the average length of each sentence, we used the Python NLTK library, which has an implementation of the Punkt sentence tokenizer, which splits a string of text into individual sentences in a more sophisticated manner than just splitting on periods. Each sentence was then fed into an NLTK implementation of the Penn Treebank word tokenizer. We then counted the total words and averaged them over the number of sentences. We also used this data to determine the length of each word, ignoring any single characters found that weren't either "a", "A", "i", or "I", to remove punctuation from this count, and averaged this over the total number of words in the document. Determining the average number of commas, semicolons, colons, and quotation marks was straightforward and just involved the implementation of the Python Counter class and counting the instances of the respective ',', ';', ':', and '"' characters. To find the average rate of conjunctions per sentence, we again used the Penn Treebank and counted the number of words tagged as "CC" and averaged this count over the course of the entire document.

To find the percentile positions of each subject and verb, we used a Java implementation of the Stanford Parser to generate a parse tree for each sentence. For each sentence, the parser tagged one or multiple words pairs as "nsubj" followed by the predicate of the argument, its position in the sentence, and the subject of the argument and its position. For each of these pairs, the percentile position of its location in the sentence was found by dividing by the total length of the sentence, which was saved and averaged over the course of the document. The max depth and total number of nodes were also found using this parse tree and again averaged over the entire document.

Another feature analyzes the text to first ascertain whether a sentence is active and passive and then loops through the entire text to compute the average use of passive sentences and the average percentage use of active sentences per given text. This is completed with a python script and nltk. The sentence is first tagged with the highly trained Stanford Parser and then focuses on the verbs. A list of tags from a sentence is given and returns the sentence if it is marked passive and in this case if there is a verb followed by a nongerund it's marked as passive.

For example, a the sample input file (sample.txt) with four sentences states:
My name is Bob.
The cat was given a dog.
The dog goes to the store.
The dog was allowed to play by his owner.

The function find_and_print_passives('sample.txt') returns an array of the two passive sentences as ['The cat was given a dog.', 'The dog was allowed to play by his owner.'] as the two other sentences are not identified as passive and thus not printed.

Additionally, we checked to see if word entities presented in texts were foreign words or not, specifically if they were in the wordnet module maintained in NLTK. Wordnet is a lexical database of English words. There are some known issues with using Wordnet such as the definition order may not be listed by most common usage and load time can be slower than other dictionaries.

Finally, we measured the average relative frequency of words with different etymologies. In particular, we identified three potential origins for

words - Latin, Germanic, and Slavic. To track down the etymologies of each word in a document, we built a web crawler to search a few websites that should be expected to have the etymology for the word. If this value was not one of Latin, Germanic, or Slavik, we interpreted the more modern language to one of those values, so that if the found etymology was English, the word would be classified as Germanic, or if it were Spanish, it would be classified as Latin, etc.To save time with words that appeared more than once in our training data, each time a word's etymology was found, it was added to a database so we would not have to spider the web again. Using these methods, we found etymologies for most of the words in our training data, which were recorded as total percentages for each paper.

With each of these features, we then used a Support Vector Machine (SVM) to interpret the results: we created a different SVM for each feature so as to identify the ability of each feature to infer the authorship of a paper. We then used an SVM of all available features to test the overall ability of our project to infer authorship. Finally, we used cross validation to make up for the fact that we have less data than we would like. 10-fold cross validation, the kind we used, helps us maximize our available data for training and testing.

# 4 Results

We tested 10-fold cross validation on the individual features as well as an aggregate SVM looking at all of the features. The results and explanations are as follows:

All features combined
Total accuracy: 82.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 19 | 3 |
| Guessed Lindsey | 6 | 22 |

This was the result of using an SVM on all of the features below at once, letting the SVM figure out the weighting. This is not the best result return, which it theoretically should be, and we suspect it is caused by misleading features (average subject position, average verb position, and conjunction usage).

Sentence length
Total accuracy: 86.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 23 | 5 |
| Guessed Lindsey | 2 | 20 |

Sentence length turned out to be a good determiner for distinguishing between Ari's and Lindsey's styles. It was better for identifying Ari's works than Lindsey's; we don't know quite what this means.

Word length
Total accuracy: 80.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 18 | 3 |
| Guessed Lindsey | 7 | 22 |

Word length was not a great determiner, but it was still good. Ari's works were less easily identified, which suggests that his word lengths has a higher standard deviation than Lindsey's.

Punctuation usage
Total accuracy: 82.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 22 | 6 |
| Guessed Lindsey | 3 | 19 |

Punctuation consisted of commas, semicolons, colons, and quotations per sentence. Ari's works generally exhibit fewer commas, for instance, and Lindsey's have noticeably more. Jordan, Jen, and Lindsey all use at least 1.2 commas per sentence, on average, suggesting that they use compound and more complex sentences. Ari and Ezra, on the other hand, use on average 0.67 and 0.79 commas per sentence. Jordan's work, though untested in these listed results, exhibits no colon usage. This type of characteristic punctuation use reflects a person's speech and thought patterns.

Conjunction usage
Total accuracy: 66.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 18 | 10 |
| Guessed Lindsey | 7 | 15 |

This was a moderately poor determiner, though it was poorer with Lindsey's papers.

Average subject position
Total accuracy: 46.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 13 | 15 |
| Guessed Lindsey | 12 | 10 |

This may as well have been random guessing, which is to be expected with an SVO language such as English. It is still nonetheless interesting to note that, despite the many freedoms that English affords us, subject and verb position as a percentile remain the same.

Average verb position
Total accuracy: 46.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 9 | 11 |
| Guessed Lindsey | 16 | 14 |

The comment above applies here as well.

Average max sentence depth
Total accuracy: 74.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 20 | 8 |
| Guessed Lindsey | 5 | 17 |

This was a test of the depth of a sentence when viewed as a dependency tree. The idea here was to identify more complex sentences. It looks as though while there is a difference, it is not as pronounced as it could be.

Average nodes
Total accuracy: 84.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 23 | 6 |
| Guessed Lindsey | 2 | 19 |

This is how many nodes there are in a sentence, on average, which roughly equates to how many words there are in a sentence. This is redundant, but as it delivers good results, it is worth keeping if only to emphasize the good choices that sentence length makes.

Etymologies
Total accuracy: 74%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 20 | 8 |
| Guessed Lindsey | 5 | 17 |

This was a collection of the percentage of words from Germanic origin, Latin origin, and unknown origin. Words were looked up by parsing the etymology sections of EtymOnline and Wiktionary, looking at which language was mentioned the most, and then marking the word as appropriate. If a word was not in either of those sites, its etymology was requested from Google following the same approach. Unfortunately, this web spider was not foolproof: many words that should have been clearly one class or the other were marked as unknown because of the hobgoblin of automation (mere mistakes in parsing). This is easily corrected, although we did not do that; it was reasonably correct enough for our purposes.

Most features combined
Total accuracy: 92.00%

| | Is Ari | Is Lindsey |
|---|---|---|
| Guessed Ari | 24 | 3 |
| Guessed Lindsey | 1 | 22 |

This was a combination of most features, excluding conjunction usage, average subject position, and average verb position; it is good to see that the results are much better here than before.

# 5 Conclusion

In conclusion, our program was able to determine authorship of papers 90% of the time. In general, Ari's works were more correctly identified than Lindsey's, which suggests that Ari's style of writing is more consistent than Lindsey's. This is aided by the fact that Ari's works were almost entirely creative writing, whereas Lindsey's were a mix. Future work should segregate the two classes of writing. Future work could also use a "committee" of SVMs to decide on the overall choice of classification so as to give greater weight to the features that are better determiners.

## References

[1] Efstathios Stamatatos *Ensemble-based author identification using character N-grams* 2006: Proceedings of the 3rd International Workshop on Text-based Information Retrieval.