Augmenting n-gram Based Authorship Attribution with Neural Networks

John-Paul Verkamp Michael Wollowski Maki Hirotani

September 7, 2010

Abstract

While using statistical methods to determine authorship attribution is not a new idea and neural networks have been applied to a number of statistical problems, the two have not often been used together. We show that the use of artificial neural networks, specifically self-organizing maps, combined with n-grams provides a success rate on the order of previous work with purely statistical methods. Using a collection of documents including the works of Shakespeare, William Blake, and the King James Version of the Bible, we were able to demonstrate classification of documents into individual groups. Further experiments with The Federalist Papers exposed potential problems with the algorithm. Finally, first exchanging n-gram frequencies with word frequencies and then exchanging self-organizing maps with k-means clustering shows that it is the combination of the two factors contributing to the algorithms success.

Introduction

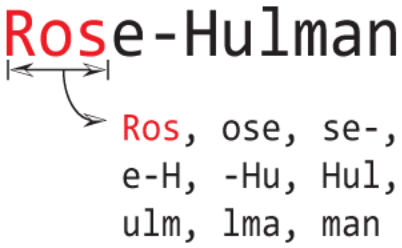
Throughout many fields and disciplines, there exists the need to quickly and accurately determine the class into which a given object belongs. One specific subsection of this problem is that of authorship attribution of written works. Authorship attribution is characterized by the ability to analyze an anonymous text or one of uncertain authorship and determine the identity of the author [9].

Two examples most clearly outline the benefits of authorship attribution algorithms: the works of William Shakespeare and the Federalist papers. Shakespeare is known as one of the most prolific playwrights of all time with nearly 40 plays credited to his name; however, some argue that all of the aforementioned could not possibly have been written by the same author in one lifetime. In such situations, statistical algorithms have been employed to give a relatively accurate estimate of just how likely it is that the each of the individual works was written by the same author as the group as a whole.

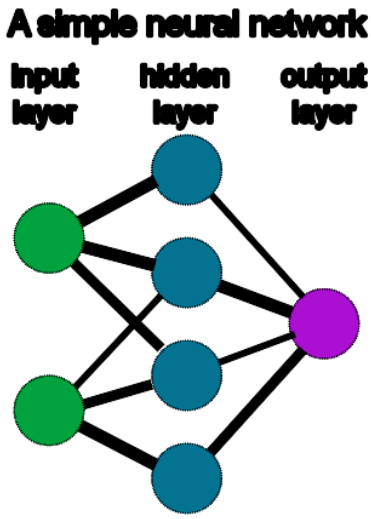
The Federalist Papers, on the other hand, are a collection of 85 articles from the time of and debating the merits of the United States Constitution. At the time that the articles were written, their authorship was a closely guarded secret; however, at the present time most believe that they were written by Alexander Hamilton, James Madison, and John Jay (in order of decreasing number of articles) [5]. In a paper by Kjell and Frieder from 1992, 2-grams were used to analyze the Federalist Papers, confirming much of the previous academic work in the field and proposing attribution for 11 previously uncertain papers [8].

Another somewhat different use of authorship attribution is to detect plagiarism, most often in the works of students or others with large bodies of written work. Since the strength of most authorship attribution algorithms rely on a large body of work on which to draw, it is rather difficult to use this method to detect whether a single paper is plagiarized without further information–although there are other algorithms that may be able to accomplish such a task with access to online databases [7]. However, with a large body to work with, it is possible to apply the same idea as to Shakespeare wherein if a single paper is not written in a similar enough style to the body of works, it can be flagged for further investigation as possible plagiarism [2].

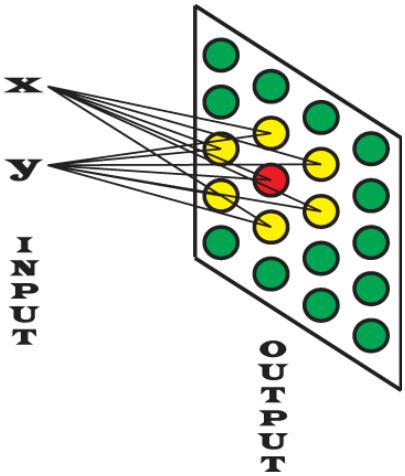
Our particular contribution to the field of authorship attribution will be the addition of neural networks to the standard use of n-grams. While n-grams by themselves provide surprisingly accurate results for most data sets, there is room for improvement. The earliest work with n-grams comes from 1976 [4] and has since been developed to the point where 95 percent accuracy is not unreasonable under certain conditions [6]. We believe that the pattern learning and matching capabilities of neural networks applied to the statistical analysis of n-grams will improve the results more than n-grams alone.



(a) n-grams



(b) Artificial Neural Network



(c) Self-Organizing Map

1 Methodology

Two primary components make up the bulk of my research in this field: n-grams and neural networks–each defined below. N-grams will be used for basic statistical analysis and provide the framework to which the neural networks will be applied. The neural networks will apply machine learning and pattern matching to the n-gram data beyond the statistical models traditionally used with n-grams.

1.1 n-grams

The first component of an augmented n-gram authorship attribution algorithm is the collection of n-grams [4]. An n-gram is defined as a set of contiguous symbols of a larger such group. For example, the string "rose-hulman" would have the following 5-grams: "rose-", "ose-h", "se-hu", "e-hul", "-hulm", "hulma", and "ulman" as shown in Figure 1a. Traditional work applying n-grams to determine authorship rely on statistical analysis to determine the most common n-grams used by an author and have revealed surprisingly accurate results using n-grams of size 4-8[6].

One particular benefit of n-grams over letter or word based frequency analysis is the collection of word order where n-grams are split between words. For example, the 5-grams for "the quick brown fox" include "the q", "ck br", and "n fox". Thus, not only is word choice encoded but also sequence information about them.

In addition to their uses in authorship attribution, 1-grams (letter frequency) and 2-grams (bi-grams) have frequently been used in other fields such as text compression [3] and data mining [14].

1.2 Artificial Neural Networks

An artificial neural network is designed to represent a simplified model of a brain in that a set of inputs are applied to several layers of virtual neurons such that the pattern in which they are activated provides output. Such networks are useful both for their ability to "learn" patterns without being explicitly programmed and also being able to store more complex data related to the state of whatever input data is being used. First proposed by Warren McCulloch and Walter Pitts in 1943, neural networks have a relatively long history in artificial intelligence [10].

Essentially an artificial neural network is a non-linear function with the ability to be trained to better match the input. Figure 1b shows a simple neural network.

The leftmost layer is the input layer to which the input vector is applied. The second, hidden layer introduces the non-linear element into the neural network. The final layer is the output of the function. Another way to model a

neural network is a pair of matrix multiplications. With an input size of n, a hidden layer with m neurons, and y outputs, the input will be multiplied by an $nx \times m$ matrix and then an $m \times y$ matrix.

Using neural networks to match patterns has a long and successful history with abilities such as function approximation, regression analysis, filtering, and clustering [12]. Thus, when applied to n-gram based frequency analysis, it is entirely possible that neural networks will boost the accuracy of the previous algorithms even further.

1.3 Self-Organizing Maps

Self-organizing maps (also known as Kohonen maps)are a subset of artificial neural networks used to produce a discrete, low-dimensional version of a high-dimensional input while maintaining the relationships between vectors with similar properties. A sample self-organizing map is shown in Figure 1c.

Similarly to a neural network, a self-organizing map is made primarily of individual components, called either nodes or neurons. Each node has a vector (of the same dimensions as the input vectors) associated with it. Generally, these initial nodes are either randomly generated or taken from a training set.

When training a self-organizing map, a competitive algorithm is used. Each input value is applied to each output node and associated with the most similar (using a measure such as Euclidean distance). Then that node and the nodes adjacent to it are trained to better match the input vector. Thus the network as a whole moves toward its final state.

For more information on how both n-grams and self-organizing maps are used in our algorithm, see the Discussion section.

2 Algorithm

The algorithm we developed consists of two components: N-grams are generated and used to calculate frequency vectors. Frequency vectors are used as input to a self-organizing map which organizes them into a two-dimensional output space.

2.1 Generation of n-grams

The first step of the algorithm is to generate a vector for each document based on the frequencies of the n-grams. The algorithm proceeds as follows:

1. Generate all of the n-grams for each document.
2. Calculate the frequency of each n-gram.
3. Choose a set of the most common n-grams from all of the documents.
4. Using the same list for each document, generate a vector of frequencies.
5. Normalize the vectors.

The first two steps are self-explanatory. As a result, each document has a collection of the most common n-grams associated with it. The next step is designed to eliminate words particular to a document. For example, in the *The Tragedy of Macbeth* the name "Macbeth" is unusually common. As such, the n-gram `_MACB` would appear frequently. However, it does not appear in any of the other documents. To eliminate outliers like this, the collection of n-grams is merged to form a list common to all of the documents. Using a set of most common n-grams from this unified list, a vector of frequencies is generated for each document. Most of the entries in this vector are near zero. To aid in the processing of this data in later stages, the vectors are then normalized.

The following table shows the first portion of the n-gram vector generated for *The Tragedy of Macbeth* (before normalization).

As expected, common words make up the most frequent n-grams. However, n-grams also capture words that appear together. This can be seen in row seven. In this particular example, words starting with "THE" follow words ending with "E" often enough to appear in the top ten most common n-grams.

Table 1: Sample n-gram frequencies

n-gram	frequency
THE	0.00765
AND	0.00591
_THAT	0.00248
THAT_	0.00238
YOU	0.00212
_WITH	0.00210
E_THE	0.00192
NOT	0.00172
WITH_	0.00162
HIS	0.00152

2.2 Application of self-organizing map

Following the generation of the normalized n-gram vectors, they are fed into a self-organizing map, a Kohonen network. Each vector is applied to each node, training the nodes in the neighborhood of the best match as aforementioned. The sum of the movement at each iteration becomes the error measurement for that iterations. This process is continued until the error falls below a certain threshold.

3 Results

Overall, the results have been extremely positive. For the primary test set including Shakespeare, Blake, and the Bible. the groups of documents are easily recognizable with few edge cases. The second test was less successful, but simple parameter tuning has shown to increase performance with future work able to continue improving on the results.

3.1 Basic results

Below is a sample run of the algorithm showing key iterations. Red documents are known to us but not the algorithm to be written by William Shakespeare, blue documents by William Blake, and green documents are the books of the King James Bible. Each was chosen because of the relatively large body of works and the ease in acquiring clean digital copies of the works.

As shown in Figure **??**, the algorithm starts with a random distribution of node weights and assigns the documents to the nearest node. Due to the random nature of the algorithm, many documents are clustered around the same nodes. The initial error value is 1.13495.

After a single iteration (as shown in Figure **??**), the error has fallen to 0.52519 and groupings are already starting to appear with the bible in the lower left, Blake in the lower right, and Shakespeare in the top right.

After another iteration as shown in Figure **??**, the groups are more well defined and the outliers are starting to return to their clusters. The most obvious remaining counterpoint of this example run is *The Shepard* from William Blake's *Songs of Innocence*. The current error has again been halved to 0.28401.

After one more iteration, the current error is only 0.10746 and the document clusters continue to grow more clear. After this step (shown in Figure **??**), the algorithm runs for a few iterations without much change. The final result for this particular run is not much different.

The final result (with an error below the threshold of 0.00001) is shown in Figure **??**. *The Shepard* is still being placed with the Bible while one of the Bible's books is between Blake's work and Shakespeare's. Because of the random initial conditions, each run through is different and produces slightly different oddities. One constant document of interest is Shakespeare's *Tragedy of Macbeth* which will be discussed later.

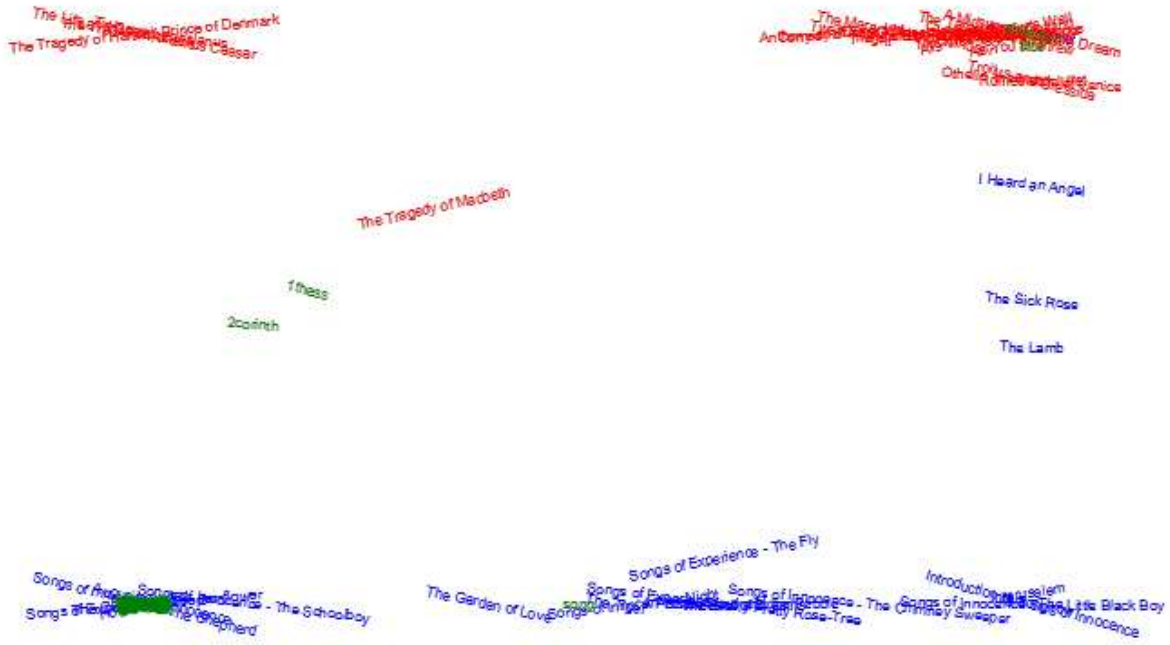


Figure 1: Sample Run - Initial Conditions

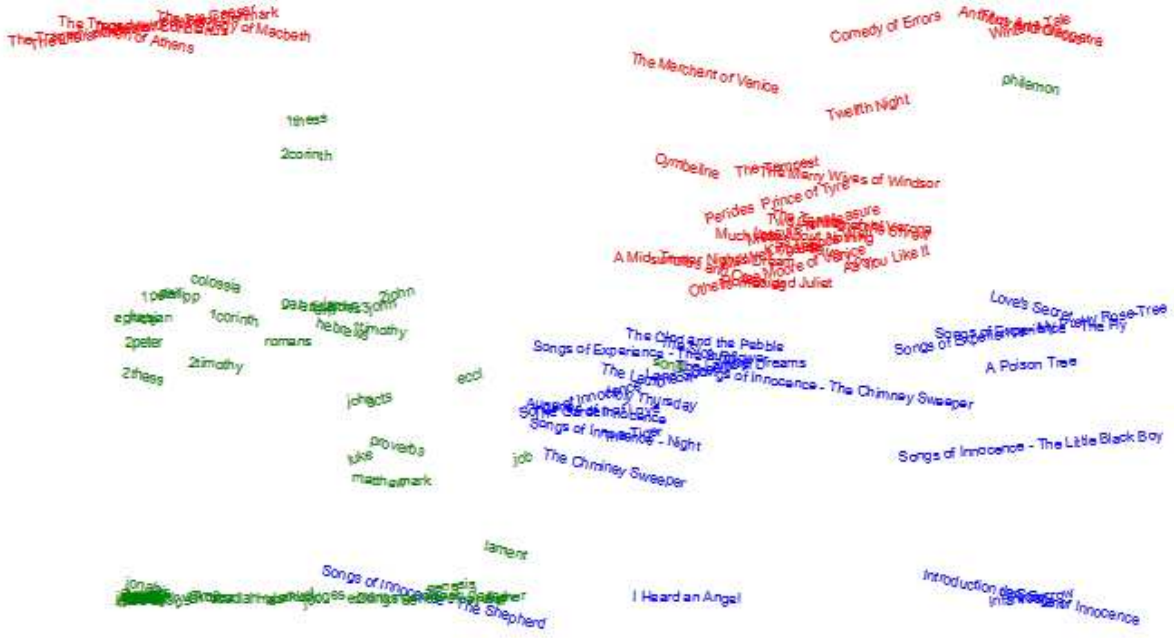


Figure 2: Sample Run - First Iterations

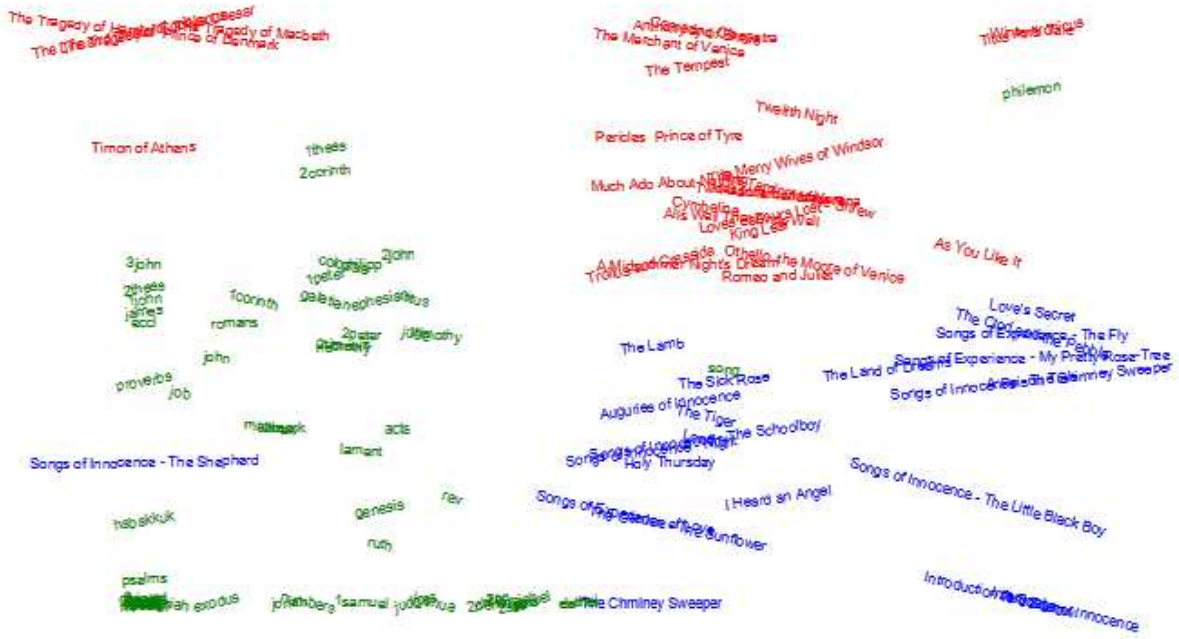


Figure 3: Sample Run - Second Iteration

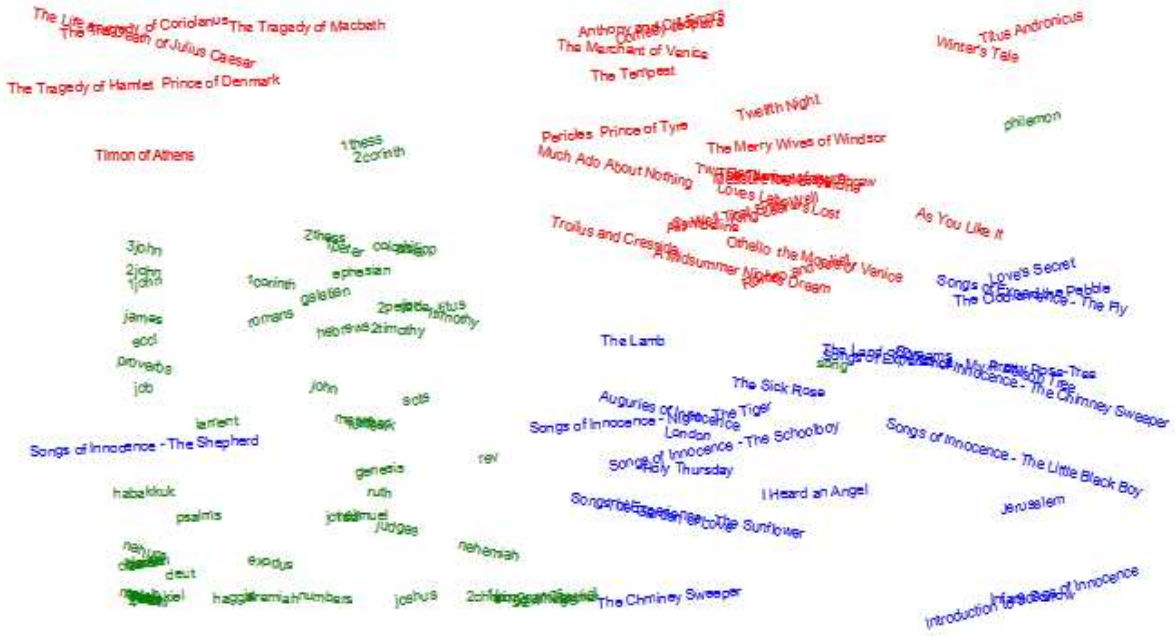


Figure 4: Sample Run - Third Iteration

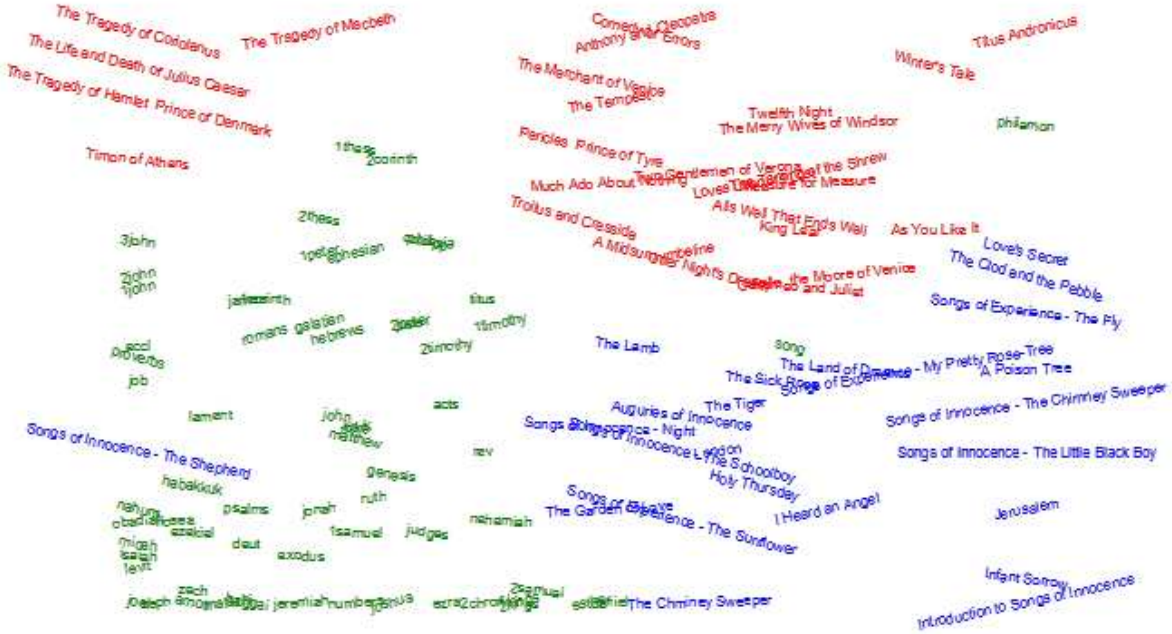


Figure 5: Sample Run - Final State

3.2 Parameter Tuning

Within this algorithm, there are three main parameters that can be chosen for the algorithm: n-gram length, frequency vector dimension, and self-organizing map grid size.

At the lower end, using 1-grams is exactly the same as letter frequency analysis and does not have much success at differentiating between documents. 2 and 3 grams were nearly the same, with little to differentiate themselves from random results; however, 4-grams are the threshold of successful document classification. Most likely, this is because 1-3 grams fail to capture the consecutive word usage that makes n-grams so successful.

After 6-grams, performance again begins to degrade. At this level, many of the most common n-grams (particularly in Shakespeare's plays) began to be the names of characters and overall frequencies were much lower and nearly level. The algorithm described above used 5-grams for it's analysis.

The next two factors were the length of the frequency vector and the size of the self-organizing maps. For these cases, the limiting factors were time and memory. Due to the nature of the self-organizing maps, increasing the size of the frequency vectors or the size of the map itself would greatly increase the run time of the program and the required memory. For this experiment, the frequency vectors contained 100 elements and the self-organizing map was 50 by 50.

3.3 The Tragedy of Macbeth

On interesting case from which to look further is *The Tragedy of Macbeth*. Unusual among Shakespeare's plays in several ways, it is not only among the shortest, it is also notably heavy in action during the first portion, and unusually light on character development (with the exception of Macbeth himself) throughout. In addition, two songs from a Thomas Middleton play are known to have been added to the play along with more conjectured to have been added. Specifically act III, scene V and act IV, scene I are thought to have been originally written by Middleton.

If this were to be the case, than the constant state of being outlier, as shown in Figure **??** would be due to the inclusion of an additional author's work. To test this theory, the noted sections were removed and the algorithm was run several times. One such run is shown in Figure **??** (the new Macbeth is shown in black). During none of the runs was the new version of Macbeth outside of the Shakespeare cluster, thus supporting the hypothesis that Middleton (or another author) added those sections to Macbeth and had a statistically significant difference in writing style.

3.4 The Federalist Papers

Another test of this method would be to determine whether correct authorship can be determined in *The Federalist Papers*. Written in support of ratification of the United States Constitution between 1787 and 1788, the original identity of the authors was unknown when they were published, although Alexander Hamilton, James Madison, and John Jay were correctly theorized. After Hamilton passed away in 1804, a document asserting his authorship of more than two thirds of the works was released. Unfortunately, at least 11 of them were more likely to have actually been written my Madison.

In 1944, D. Adair first published his theory of ownership of the documents [1], later supported by a computer's analysis by F. Mosteller and D.L. Wallance in 1964 [11].

Unfortunately, the original run of my algorithm failed to make a distinction between the authors as shown in Figure **??**. In that figure, Hamilton's papers are shown in green, Madison's in red, Jay's in blue, and the papers on which there is a conflict are shown in black.

Theorizing that the relatively complex language and longer average word length might have an effect on the results, we increased the n-Gram size from 4 to 6 and decreased the length of the frequency vector. Figure **??** shows the results. While still not as perfectly segmented as the initial tests, all of Jay's papers were clustered and there is a distinct cluster of the unknown papers in the upper left corner.

While these particular results are not conclusive, I believe that with more time and effort, it would be possible to improve the clustering even further.

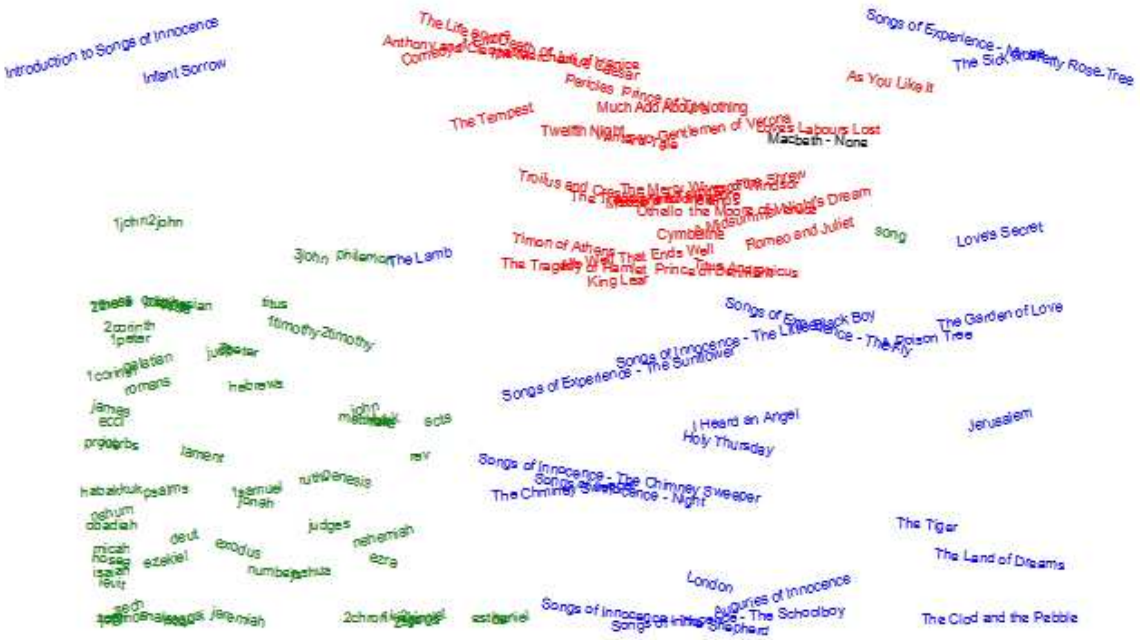


Figure 6: Analysis of Trimmed Macbeth

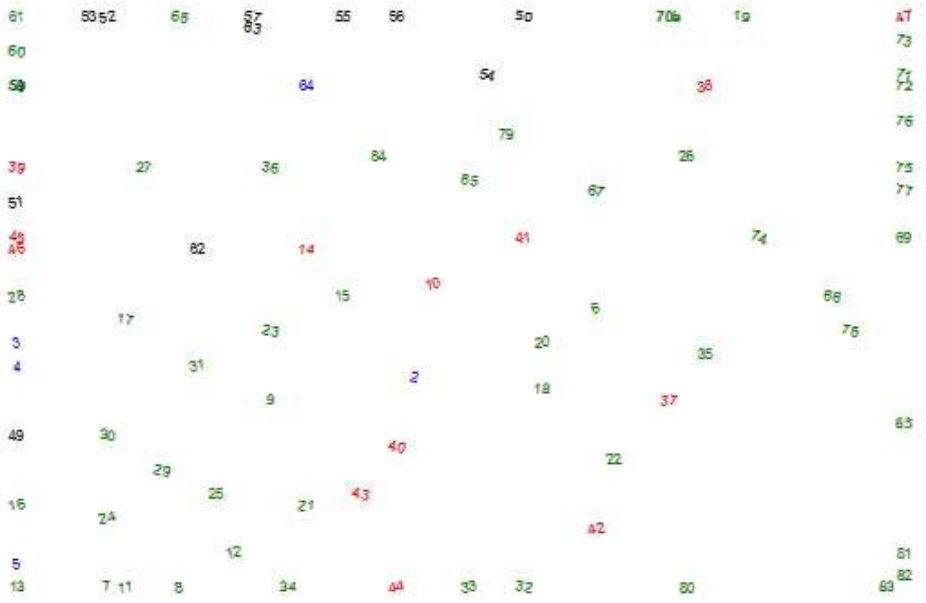


Figure 7: The Federalist Papers - First Run

Figure 8: The Federalist Papers - Second Run

4 Discussion

To study in detail the power of chosen combination of n-grams and self-organizing maps, several control experiments were conducted in which each component was tested with a nearly equivalent alternative. We tested the following combinations: n-grams and feed forward networks, n-gram with k-means clustering, and word-frequencies with self-organizing maps.

4.1 Use of N-grams and Feed-Forward Networks

In this version, the sliding window used to generate the n-grams is used as input to a feed-forward network. A network is trained on a collection of documents written by the same author. A document about which we wish to learn authorship is presented to the feed network in the same manner as it is trained. A single final value is returned indicating the degree to which the input matches the trained neural network.

This approach proved to be poor with seemingly random results even after several training cycles. Feed-forward networks have been used successfully to learn pronunciation for a given sequence of letters such as in NETtalk[13]. While NETtalk used a sliding window of characters similar to n-grams, the goal was to associate a given phoneme with a sequence of characters. This is a considerably different problem to ours, where we only have binary information about the given input, whether it was a member of a set or not.

4.2 Use of N-Grams with K-Means Clustering

In this experiment, we replaced the self-organizing maps with k-means clustering. K-means clustering starts with a set of randomly initialized clusters. K inputs are selected as means. At each iteration, each input is assigned to the nearest mean and then the mean of each cluster is moved to the average of it's input. This continues until either a set number of iterations have passed or the means have reached a steady state.

We experimented with different values of k, from as low as 3 to as high as 50. The most successful run used 5 clusters and correctly identified roughly half of William Blake's works as being separate from Shakespeare's or the Bible's. However, all of the remaining documents were assigned to the same mean with no differentiation between them.

The problem with k-means clustering seems to be due to the similarity of the values within the frequency vectors. K-means clustering tends to work more efficiently with more broadly spread means as more narrowly spread means tend to cause their associated clusters to converge more quickly.

4.3 Use of Word Frequencies with Self-Organizing Maps

For this experiment, we used word frequencies instead of n-grams. As shown in Figure **??**, the results are far less impressive than using n-grams. Shakespeare's works are still separate; however, the Bible and William Blake have clustered together.

We attribute these results to the similar styles by Blake and the Bible as opposed to the more complex, flowery prose favored by Shakespeare. While in this approach, we gain the ability to recognize a wider variety of words, this seems to be at the expense of losing sequence information between words.

5 Future Work

A number of possible options are available for future work, including testing with larger data sets, training data versus testing data, plagiarism detection, and the effect of foreign language textbooks on writing style.

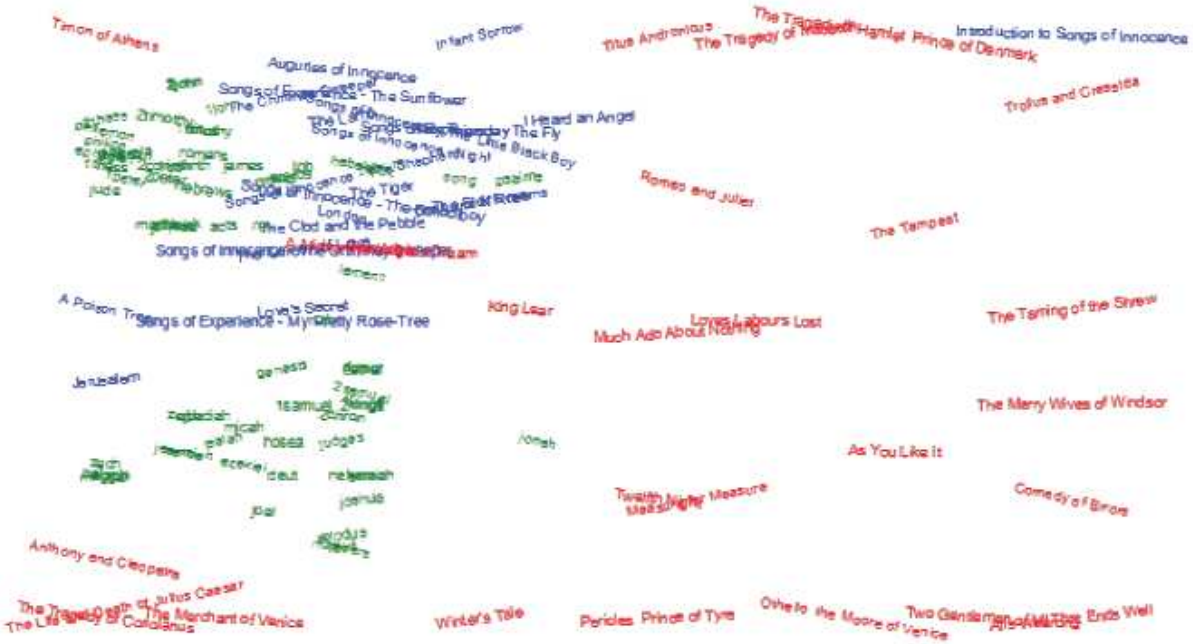


Figure 9: Word Frequencies with Self-Organizing Maps

5.1 Larger data sets

At the present time, the largest data set that has been tested with the algorithm contains 240 documents (using Shakespeare, Blake, and the Bible). For those runs, we used 5-grams, a frequency vector with 100 dimensions, and a 50 by 50 self-organizing map. At that scale, each iteration takes approximately 2-3 minutes to run. With an average run taking 10 iterations to converge, each run requires 20-30 minutes.

One useful future application of the algorithm would be to vastly expand the scope of each of the parameters to determine which are the most important to its relative success and to determine if the results are scalable to considerably larger data sets.

5.2 Training versus testing

One problem with the current algorithm is that there is no separation of testing and training data. The entire purpose of the algorithm revolves around a single run which starts with a mixed collection of documents and partitions them into clusters. However, no work has yet been done to determine if these same results generalize to documents added to the collection at a later time. It is possible that only the documents first fed into the algorithm are successfully classified; however, it's likely that additional documents will be classified as well.

5.3 Plagiarism detection

A potential real world application of authorship attribution is to detect plagiarism. Previous work in this field generally uses a large online database of known documents and compares a new document against this library to look for a match. Our algorithm would add a suspect document to a collection of documents written by the same author to check for consistency in style and as such ascertain the likelihood of plagiarism. If the suspect document matches the writing style of the other documents in the collection, it will be included in the cluster; if not, it will be an outlier. There is an exception to the success of this method, namely if an author consistently were to plagiarizes from the same source.

5.4 Effect of writing courses on style

Another interesting application, similar to the one above would be to determined the effect of writing courses. Presumably, if a student improves their writing style, their new works will be outside of the cluster of their works collected before taking a writing course.

6 Conclusion

Overall, the results using n-grams and self-organizing maps have been extremely positive, successfully clustering documents into groups with only the n-gram frequency information as input. With a variety of avenues in which to proceed, this algorithm has a lot of potential.

References

[1] D. Adair. The Authorship of the Disputed Federalist Papers: Part II. *The William and Mary Quarterly: Magazine of Early American History, Institutions, and Culture*, pages 235–264, 1944.

[2] A. Barrón-Cedeno and P. Rosso. On Automatic Plagiarism Detection based on n-grams Comparison. *Advances in Information Retrieval*, pages 696–700.

[3] T.C. Bell, J.G. Cleary, and I.H. Witten. *Text compression*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1990.

[4] W.R. Bennett. *Scientific and engineering problem-solving with the computer*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1976.

[5] DI Holmes and RS Forsyth. The Federalist revisited: New directions in authorship attribution. *Literary and Linguistic Computing*, 10(2):111, 1995.

[6] V. Keselj, F. Peng, N. Cercone, and C. Thomas. N-gram-based author profiles for authorship attribution. In *Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING*, volume 3, pages 255–264. Citeseer, 2003.

[7] D.V. Khmelev and W.J. Teahan. A repetition based measure for verification of text collections and for text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 104–110. ACM, 2003.

[8] B. Kjell and O. Frieder. Visualization of literary style. In *IEEE International Conference on Systems, Man and Cybernetics, 1992.*, pages 656–661, 1992.

[9] H. Love. *Attributing authorship: an introduction*. Cambridge Univ Pr, 2002.

[10] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[11] F. Mosteller and D.L. Wallace. *Inference and disputed authorship: The Federalist*. Addison-Wesley, 1964.

[12] B.D. Ripley. *Pattern recognition and neural networks*. Cambridge Univ Pr, 2008.

[13] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1(1):145–168, 1987.

[14] I.H. Witten, Z. Bray, M. Mahoui, and B. Teahan. Text mining: A new frontier for lossless compression. In *dcc*, page 198. Published by the IEEE Computer Society, 1999.