Intelligent Authorship Confirmation

COSC 410: Final Project

Matthew Sobocinski and Brady Sheehan
Department of Mathematics and Computer Science
Duquesne University, Pittsburgh, PA, USA

April 25, 2016

1 Introduction

Given a document and an author's stylometric profile, can we determine if that author wrote the document? Solving this problem is known as authorship verification. The task of identifying the author of a work through the use of stylometry has been around for hundreds of years. The process of building an author's profile based on their writing style is known as stylometry. Much like a fingerprint, samples composed by an author can be used to identify unique characteristics about their writing style. These features can then help in ultimately identifying who wrote a work. However, using the profiles of an author to distinguish one author's work from another author's work is challenging due to the large amount of noisy data. Advances in natural language processing and artificial intelligence have given researchers and practitioners new techniques for extracting and identifying patterns in this data. Ultimately, these techniques will allow us to verify whether or not a given text was written by a particular author.

In this work, we attempted to solve the authorship verification problem with cross-topic and cross-genre documents using a supervised feed-forward neural network. To introduce our work, we will first present background about the authorship verification problem and why an artificial intelligence approach is being considered. Next, we will explain what features were chosen to create a stylometric profile for a given author, as well as the reasons those features were considered. We will then introduce the neural networks used for processing these features and explain how we tested them. The use of these features with various network configurations and their effectiveness at solving the authorship verification problem will be compared. Finally, we will discuss results of our work, what we have learned, and where future work on this project is headed.

2 Background

Verifying the author of a document is an important problem with a multitude of practical applications across a wide variety of disciplines. This problem is essentially one of classification: can we classify an unknown document into the set of known works by a given author? Examples of such applications include course instructors checking for plagiarism, reporters and journalists attempting to authenticate a document, or even law professionals disputing a will.

Natural language processing, commonly abbreviated as NLP, is a sub-field of artificial intelligence that uses computer systems to parse and analyze human language, either in spoken or written form. As computing power, software, and techniques improve, the ability of computers to understand human languages has increased greatly. Furthermore, as more information is digitized, the scope of areas to which language processing can be applied increases on a daily basis. Prior to these technological advances, analyzing documents to perform authorship identification was challenging and time intensive.

For some language processing problems related to authorship, it is sufficient to parse through the documents of interest, extract features, and compare them with a simple statistical analysis. However, some of the methods for authorship identification that have not relied on techniques from artificial intelligence, such as neural networks, have struggled to obtain accuracy as good as the results obtained by neural networks [1]. In [1], they were able to obtain a correct classification rate between 80% and 90% when differentiating works written by two poets.

Generating a large amount of features given a text is relatively easy. The use of artificial intelligence by way of neural networks offers major advantages toward gaining the necessary insight from these features. In particular, they are great at working with noisy data and can usually generalize to unseen inputs well.

Much research in the natural language processing field has been dedicated to attributing a work to an author, where the set of authors under consideration is essentially unrestricted. Authorship verification, on the other hand, has been pursued much less. As a result, it is unknown if neural networks trained with features about an author could solve this problem completely.

3 Methods

Our approach to the authorship verification problem involves building a profile of a few authors through feature extraction and feeding these profiles to a neural network for training. The basic idea is to "teach" the neural network to distinguish the writing styles of different authors. Then, when given an unknown work, one could test its features against an author's features that were included in the training set. Ideally, the network could then verify if the unknown work was written by the known author.

All of the techniques we considered for this work were from the "bag of words" perspective. In this view of a document, we consider the words to be independent from one another. This framework does not attempt to identify the genre of the work or the semantics of a given sentence or section from the work. We consider the work to be a collection of unordered words or tokens.

3.1 Data Collection

The data collected for the experiment was made up of over 100 texts written by 12 Victorian era authors*. Each author in the corpus had between 2 and 31 works. It is also crucial to point out that this dataset spans multiple topics and genres.

It was important for testing purposes to ensure that the authors used lived during the same time period because such writers tend to write similarly. If this were not the case, a significant bias would be present in the experiment. This became clear when the initial feature vectors were created. For example, many works contained sentences that were over 100 words in length, a characteristic that is very atypical to modern writing styles. This disparity between eras could make it easy to differentiate between two stylometric profiles, defeating the usefulness of this experiment.

After collecting the data, these works were broken into three categories. Two authors and all of their works were entirely removed from the original dataset and placed into a category we will use for out-group testing. Then, a few works (but not all) by five authors were removed from the dataset and placed into another group for in-group testing. The dataset that remains after these works were

removed will be called the training dataset and is used in training the neural networks described here. The out-group dataset will be used to test the network's ability to correctly verify documents by authors who were not included in the training dataset. The in-group dataset will be used to test the network with authors it has been trained with but with works by those authors that it was not trained with. To see all of the works by each author and the exact details of these categories please refer to our appendix in section 7.

We expected the neural network to perform well on the in-group testing set. It seems reasonable for the network to be able to generalize and therefore correctly classify documents it has not seen before by authors it has seen. These predicted results come from the hypothesis that an author's style is similar across works and that the network would be able to understand an author's stylometric profile after training. In contrast, we predicted that the neural network would not be able to generalize to authors and works that were not in the training set. Reasoning behind this idea comes from the idea that the network may not understand the unknown author's stylometric profile. The network has not seen any documents by this author during training and could therefore incorrectly attribute it to another author seen during training.

3.2 Feature Extraction

Many features were tested, but most were deemed unhelpful and uninformative. Metrics that were recorded included most unique words, average word length, average sentence length, most common words, most common first word, sentence length frequencies, word length frequencies, and character and word n-grams of various lengths.

In order to determine whether or not an author's works should be treated as one large file in the training phase or left as individual files, some preprocessing was performed. Java code was written to parse the corpus of data and concatenate all of an author's works into a single file. All of the aforementioned features were calculated on both the individual files and the concatenated files using Python (version 2.7.*). Based on the individual metric being considered, different string processing was performed on the document that was rather cumbersome. For example, calculating average word length required the removal of punctuation. Other metrics, including most common first word, were overly sensitive to "stop-words", such as "the" or "our", which led to their removal as well.

Ultimately, it was decided to treat each file as a separate entity because doing so yielded the most unique features among authors. The features extracted from the concatenated files all converged to approximately similar values. For example, almost every author's average word length was ≈ 4.50. It was inferred that such similar values would either add a negligible amount of information to the neural network or simply add noise to the dataset.

Of all the metrics that were tested, only two were used in the final algorithm. Many were left out because they failed to yield significantly unique features. The two metrics that survived the filtering process were sentence length frequencies and word length frequencies, both of which were calculated with a process that can best be described as *binning*. For word length frequencies, 11 bins were created. Each bin corresponded to a count of the number of times that that word length was seen throughout the document. As each word in the document was processed, the bin whose number corresponded to that word's length value was incremented. Word lengths ≥ 11 were counted in the 11^{th} bin. After all of the bins were filled, the counts in each bin were divided by the total number of words to create a vector of word length frequencies for the entire text. The same process was followed

for sentence length frequencies. Instead of 11 bins, 50 were used, where the number of sentences with ≥ 50 words was maintained in the 50^{th} bin. These percentages, along with the 11 percentages of word length, formed the entirety of a work's feature vector. The use of 11 bins for word length frequencies resulted from the hypothesis that most words are not much longer than 11 characters on average. The use of 50 bins for sentence length frequencies followed from several early tests of the neural network.

3.3 Neural Network

The perceptron, one of the first neural network training algorithms, is used in conjunction with supervised binary classification. Once trained, the network makes a decision about which of two groups the testing instance belongs to. For our purposes, the network decided whether or not the pair of features concatenated in the input vector were extracted from texts written by the same author.

For this project, we used the Neural Network Toolbox Release 2015a. This toolbox includes various functions for configuring and training neural networks. After reading the documentation to understand what neural networks for classification were available, we started to familiarize ourselves with the parameters available to configure neural networks in MATLAB. Specifically, we tested our data with two different neural networks from MATLAB based around the perceptron, lvqnet and patternet, with various configurations.

The lvqnet function creates a learning vector quantization neural network that uses two layers. The network uses the first layer to cluster the input vectors. Then, the network uses the second layer to combine groups of clusters found with the first layer into classes defined by the target data. On our training data, this network did much worse than random and only had a correct classification rate of approximately 20%. This network was not considered for further testing because of its poor performance.

The other kind of network considered was the patternet. In MATLAB, patternet is a multilayer feed-foward network. Such a network processes inputs in only one direction and therefore connections between units cannot form a loop or cycle. This type of network can be trained with multiple training functions. The training functions considered for this project are listed in Table 1. Each of these training functions exhibited varying levels of success with different numbers of hidden layers.

Training Function	Technique Name
trainlm	Levenberg-Marquardt Backpropagation
trainscg	Scaled Conjugate Gradient Backpropagation
trainrp	Resilient Backpropagation
trainbr	Bayesian Regularization Backpropagation
traincgp	Conjugate Gradient Backpropagation with Polak-Ribiére Updates
traincgf	Conjugate Gradient Backpropagation with Fletcher-Reeves Updates
traincgb	Conjugate Gradient Backpropagation with Powell-Beale Restarts

Table 1: The name of each training function used with the patternet neural network in MATLAB and the corresponding technique name.

3.4 Testing

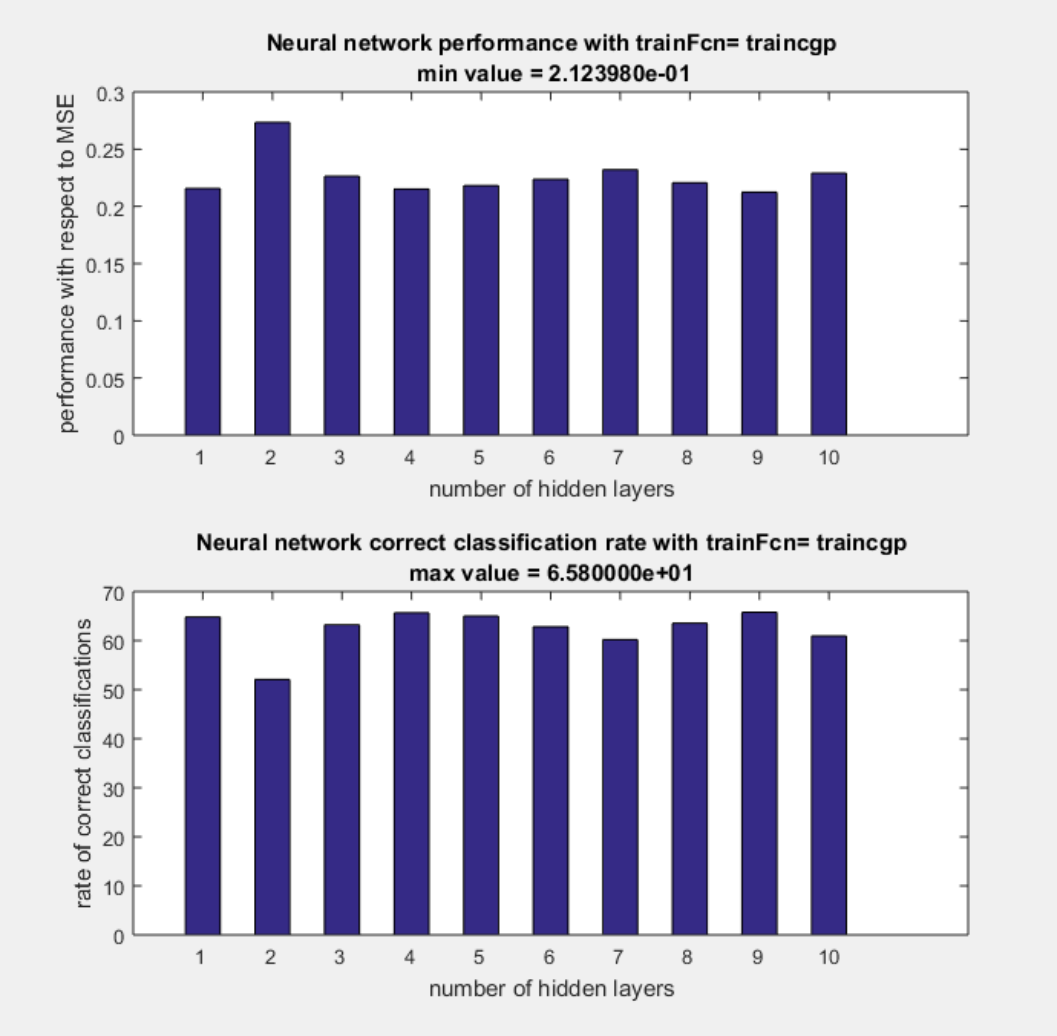

Figure 1: Network performance for the traincgp function. The x-axis corresponds to the hidden layer sizes, $10, 12, 15, 20, 25, 35, 60, 75, 100, 250$, respectively. The top bar graph shows performance for each number of hidden layers with respect to mean squared error. The bottom bar graph shows performance for each hidden layer with respect to the correct classification rate.

Since our research problem involves determining whether or not two texts were written by the same author, it was necessary to concatenate pairs of feature vectors into a single vector. This pair of features then served as a single input to the neural network. Essentially, two types of input vectors were created. The first type consisted of same author vectors, where the two feature vectors used in the concatenation were extracted from two separate works by the same author. This sample input to the neural network was given a target output value of 1. Similarly, the second type consisted of different author vectors, where the two feature vectors were extracted from works written by different authors. These input vectors were mapped to 0 in the target output file.

We then generated 2 input files for each dataset, one input file that corresponded to the vectors generated from same authors and another corresponding to the set of features generated from different

authors. For each dataset, the different input files were concatenated and the entire set of features was used as the set of samples that was fed to the neural network. Generating the data in this way ensured that there would be an equal amount of pairs in both classes and that the network would not be biased toward one class.

Now that all of the data is generated, we started testing the two neural networks in MATLAB. The training dataset was further divided randomly into three categories for the neural network. Exactly 75% of the data went into one category for training, 5% went into another category for validation, and 20% went into a third category for testing. Of these subcategories, the correct classification rate from the testing subcategory was used to select the best network configurations for further testing.

The first neural network we trained was lvqnet. We trained the lvgnet with 25 hidden layers and this process took almost 9 hours to complete. The correct classification rate from this network was approximately 20%. Due to the length of time required to train this network and early signs of its ineffectiveness with our data, we did not perform any further testing with it.

Since the other neural network, patternet, has various training functions that exhibit different characteristics, it was necessary to train each one with varying hidden layer sizes. To do this, we created a MATLAB script that would configure and train a patternet with each of the 7 training functions with hidden layer sizes between 10 and 250 layers. In this process, dozens of neural networks were created with different configurations. This script would then plot two bar graphs for each training function. One bar group would depict the neural network's performance with respect to mean squared error for each number of hidden layers and the other would depict the network's performance with respect to the correct classification rate for each number of hidden layers. This set of plots allowed us to select the best neural network configuration for each training function. An example of one of these plots for the traincgp function is given in Figure 1.

The 7 neural network configurations we selected, along with the number of neurons we found to be optimal with our training data, are shown in Table 2. After training, we tested the 7 network configurations found to be optimal on the in-group and out-group datasets. The results of these tests are also shown in Table 2.

4 Results

The expectations we had for results on this project were not entirely met. We were surprised that the in-group classification rate was much less than the out-group classification rate. Although the classification rates are not as successful as we would like them to be, the out-group rate was still, on average, better than random. In the future we would like to investigate why the out-group rate was higher than the in-group rate.

Table 2: An overview of the results from the patternet neural network with each training function using the training dataset, the in-group dataset, and the out-group dataset.

5 Conclusion and Future Work

As mentioned, the problem of authorship verification with neural networks has not yet been solved.

Training Function	Correct Classification Rate (%)			Number of Neurons
Dataset	Training	In-group	Out-group	-
trainlm	66.3	15.4	53.3	60
trainscg	65.5	15.4	52.5	60
trainrp	65.5	25.0	53.3	20
trainbr	66.5	15.4	49.2	12
traincgp	65.8	15.4	52.5	15
traincgf	65.7	15.4	52.5	100
traincgb	65.5	15.4	50.4	50

However, our tests show that solving this problem using neural networks is at least feasible and that future work has the potential to yield greater success rates.

Despite the regrettable time constraints, significant knowledge was gained in terms of how to approach this problem in the future. We found that the metrics considered here, word length frequencies and sentence length frequencies, are not enough by themselves to adequately classify an author's work using a neural network. Through this project, we learned many useful features of the Python programming language that can be applied to other projects in the future. In addition, we have become very familiar with the Neural Network Toolbox in MATLAB and neural networks in general. In completing this project, we also gained valuable experience in applying object-oriented programming techniques.

Future work on this project may focus on discovering and extracting more meaningful features, testing the feasibility of different neural network configurations, and investigating the significance of corpus size. It was shown in [4] that metrics we did not consider perform well in classifying authors and we would like to consider some of these metrics in the future.

6 References

[1] Hoorn, J.F., Frank, S.L., Kowalczyk, W., and Ham, F.V.D. (1999). "Neural network identification of poets using letter sequences". Literary and Linguistic Computing, 14(3), 311–338.

[2] Juola, Patrick. "Authorship Attribution". Hanover, MA: Now Publishers Inc., 2008. Print.

[3] MATLAB and Neural Network Toolbox Release 2015a, The MathWorks, Inc., Natick, Massachusetts, United States.

[4] Oren, H., Christian, W., and Anika, P. (2016). "Authorship verification for different languages, genres and topics". The International Journal of Digital Forensics and Incident Response archive, 16(2), S33-S43.

*We would like to thank Dr. Patrick Juola for providing the data and for helpful discussions while working on this project.

7 Appendix

7.1 Corpus Authors

Below is a list of authors whose works were used in testing and training:

1. Charlotte Bronte
2. Wilkie Collins
3. Charles Dickens
4. George Elliot (Real Name: Mary Ann Evans)
5. Thomas Hardy
6. Henry James
7. Sarah Orne Jewett
8. George Meredith
9. David Graham Phillips
10. William Makepeace Thackeray
11. Anthony Trollope
12. Edith Wharton

7.2 Additional Project Information

To see the complete list of all of the author's works and the list of stop words, visit our Github project page `https://github.com/BradySheehan/AI_Project`.