Wallace: Author Detection via Recurrent Neural Networks

Leon Yao
Department of Computer Science
Stanford University
leonyao@stanford.edu

Derrick Liu
Department of Computer Science
Stanford University
derrick.liu@stanford.edu

Abstract

Author detection or author attribution is an important field in NLP that enables us to verify the authorship of papers or novels and allows us to identify anonymous authors. In our approach to this classic problem, we attempt to classify a broad set of literary works by a large number of distinct authors using traditional and deep-learning techniques, including Multinomial Naive Bayes, linear SVMs, and Recurrent Neural Networks (RNN).

1 Introduction

Published authors often have a distinctive way of writing that is apparent to human readers. Given a small body of text, the ability to accurately identify authors can be valuable in research and other fields where consuming and synthesizing large amounts of relevant data is important, since a person interested in a small body of text by an author is likely to be interested in larger, related works written by the same author.

Author attribution or stylometry is a well known problem in NLP with many applications [3]. Previous non-statistical methods and traditional machine learning techniques have attained reasonable performance on this task [1]. Neural networks have also been used to successfully identify authors in small sets of related works [8]. In our project, we attempt to classify broad sets of literary works by many different authors by leveraging recurrent neural networks (RNNs).

2 Background

Over the last 50 years, many approaches have been used to solve this problem, including Bayesian classification, support vector machines and latent Dirichlet allocation. SVM approaches used many features such as complexity measures, function words, syntax and parts of speech, function lexical taxonomies, and n-grams. Many of the datasets used have fewer than 50 authors – with some fewer than 10 authors – comprising texts that are over 200 words [6]. Because of the range of datasets (number of authors and length of text) it is hard to have a standard accuracy to aim for. These methods already have accuracies of 80-90 percent, which makes it an uninteresting problem to improve using RNNs.

Instead, for our project, we wanted to explore if author attribution is possible when given a large number of authors with minimal amount of words per example. Given the success of of SVMs, it will be interesting to see if a RNN can learn the same features used. An RNN's ability to capture sequences of words definitely enables it to learn parts of speech, syntax, and n-grams.

3 Data

We use a cleaned dataset of 3,036 public-domain English language books written by 142 prominent authors from a sample of Project Gutenberg's library [2]. The dataset is comprised of full literary works in a plain-text format – "external" metadata (e.g. transcription notes, license information) is removed, while "internal" metadata (e.g. tables of contents, chapter headings) is retained.

For each author, we coalesce all of their works into a single document. From these documents, we create a preliminary dataset by randomly choosing up to 50 passages for each author: each text passage ranges in length from a parametrized minimum number of words to a paragraph in length. This process produces a collection of passages, each labeled with its corresponding author. We further filter these passages with various heuristics in an attempt to exclude passages consisting entirely of "internal" metadata. We generated datasets of 18-word passages and 50-word passages. Since passages with more words are less common, the 50-word dataset contains fewer passages.

	18-word	50-word
number of passages	7050	7026

3.1 GloVe Vectors

After generating our passage datasets from the Gutenberg corpus, we process these datasets to create a separate GLoVe vector representation of each passage. Because of the limited amount of data per author (an author only has so many published books), it is unreasonable to learn the word vectors from scratch. Instead, we start training using pretrained glove vectors trained on Wikipedia [5]. To create this GLoVe representation, we iterate over each passage and replace each word with its corresponding GLoVe vector. This process generates a $\texttt{num\_words} \times |\texttt{glove\_vector}|$ matrix for each passage, where $|\texttt{glove\_vector}| = 300$.

4 Approach

To assess the feasibility and performance of this task, we first implemented a baseline using traditional machine learning techniques using the publicly available *scikit-learn* machine learning library [4]. For implementing and training our neural network approach, we initially used a publicly-available recurrent neural network implementation written in Theano, but later switched to using the PyBrain neural network framework due to speed concerns, significant implementation difficulties, and overall ease-of-use [7]. The following subsections detail the models used in this project.

4.1 Baseline

For both baselines we feed in our original passages into a count vectorizer, where each word is replaced with an index. From each passage, we remove stop words ('and', 'the', etc), words with small or large document frequency and we use unigrams and bigrams. We then transform our count vector using tfidf.

For our SVM, we use a gaussian kernal, and an l2 penalty regularization. For the Multinomial Naive Bayes, we use standard model with laplace smoothing.

4.2 Recurrent neural network

A recurrent neural network is a model that allows us to input a sequence of inputs. At every step, the model will update its internal representation of the input so far, giving it a form of memory. It will also predict the label at every time step, allowing the model backpropogate the error and update its weights. The model is good for capturing sequences of words such as passages.

We implemented a recurrent neural network with a single hidden layer and softmax loss. The hidden layer size is varied in our experiments. For our RNN implementation, we exclusively train and test on the GLoVe vector representations of our dataset, with each passage being input as a sequence of GLoVe word vectors. Since we are attempting to classify whole passages, not individual words

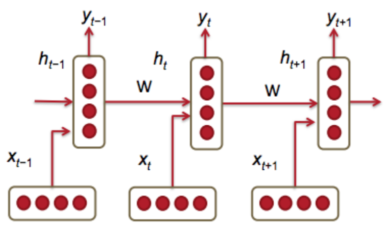

Figure 2: A Recurrent Neural Network (RNN). Three time-steps are shown.

Figure 1: Basic RNN model

$$h_t = Wf(h_{t-1}) + W^{(hx)}x_t$$

$$\hat{y} = W^{(S)}f(h_t)$$

Figure 2: RNN equations

or word sequences, we only consider the last output of the network as the true classification result. Only the last network output is used to train the network.

5 Experiments

For our baseline SVM and multinomial naive Bayes models, we performed many experiments varying the n-grams used, the min/max document frequencies, kernels, and l2 regularization constant. Only the best model's accuracies are presented in the table below.

For our RNN experiments, we performed two sets of experiments: one with the 18-word passage dataset, and one with the 50-word passage dataset. We performed exploratory hyperparameter tuning on each set of experiments, changing both hidden layer size and the learning rate of the network. For 18-word, we trained each experiment for 100 epochs; our 50-word experiments required much more time to train a single epoch, and were therefore time-limited - we list the number of epochs these experiments were trained on below.

6 Results

See figure 3 and 4 below.

		Training Accuracy	Test Accuracy
Baseline SVM 18 word		0.99	0.12
Baseline multinomial naive Bayes 18 word		0.96	0.115
Baseline SVM 50 word		1.0	0.188
Baseline multinomial naive Bayes 50 word		.99	0.187
RNN 18-word	hidden-dim 10, lr 0.05	0.087	0.044
	hidden-dim 20, lr 0.05	0.141	0.057
	hidden-dim 30, lr 0.05	0.173	0.081
	hidden-dim 40, lr 0.05	0.212	0.071
	hidden-dim 50, lr 0.1	0.219	0.066
	hidden-dim 50, lr 0.3	**0.223**	**0.095**6
	hidden-dim 100, lr 0.1	0.259	0.082
RNN 50-word	hidden-dim 10, lr 0.05 (46 epochs)	0.063	0.031
	hidden-dim 20, lr 0.05 (48 epochs)	0.119	0.063
	hidden-dim 30, lr 0.05 (48 epochs)	0.132	0.070
	hidden-dim 40, lr 0.05 (46 epochs)	0.144	0.080
	hidden-dim 50, lr 0.1 (40 epochs)	0.105	0.058
	hidden-dim 100, lr 0.1 (40 epochs)	0.139	0.081

Figure 3: Training and test results for all models. Models trained on 100 epochs unless noted.

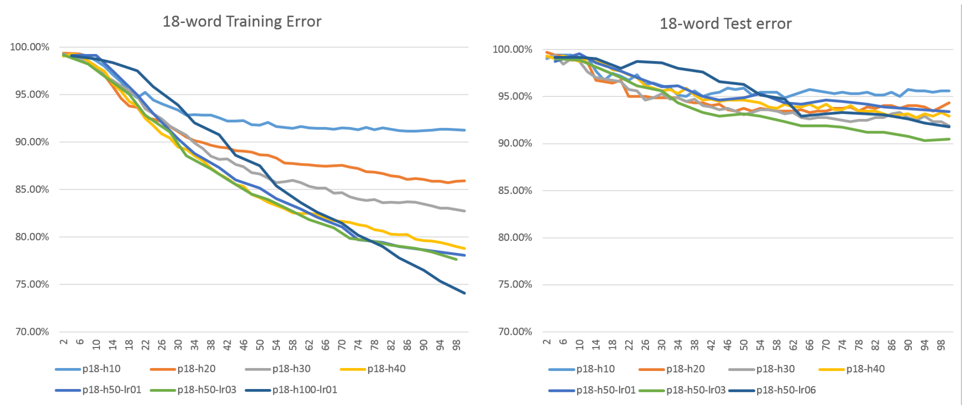

Figure 4: 18-word short passage training and test error

7 Discussion

7.1 Baseline

Both baselines have very high training accuracies for both 18-word and 50-word inputs (both above 96 percent). However, the input length heavily influences the testing accuracy. The longer our passages, the more data our model can train on, and the higher its accuracies. However, these accuracies are not even close to the accuracies achievable in the papers that use SVM on small author datasets with long passages. This shows that reducing the passage length and increasing the number of classes, makes this problem significantly harder.

Given that our classification problem has 142 separate classes, our traditional classifiers perform significantly better than random chance (which would yield an accuracy of 0.007). This implies that this author identification task is definitely feasible.

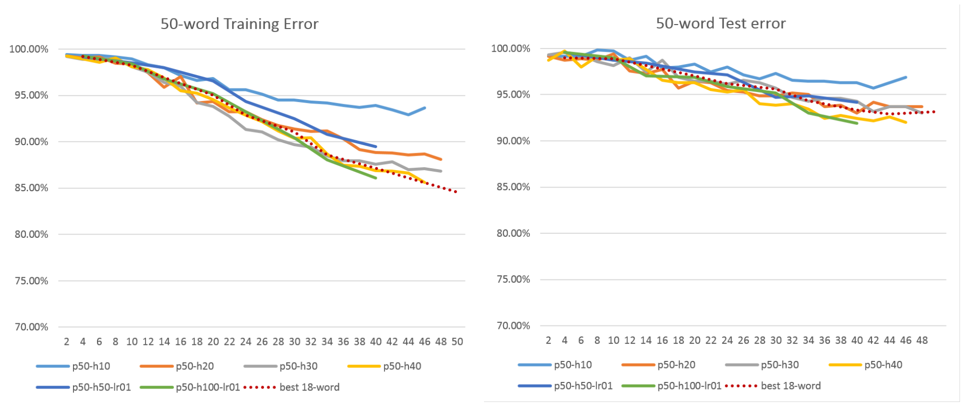

Figure 5: 50-word medium passage training and test error

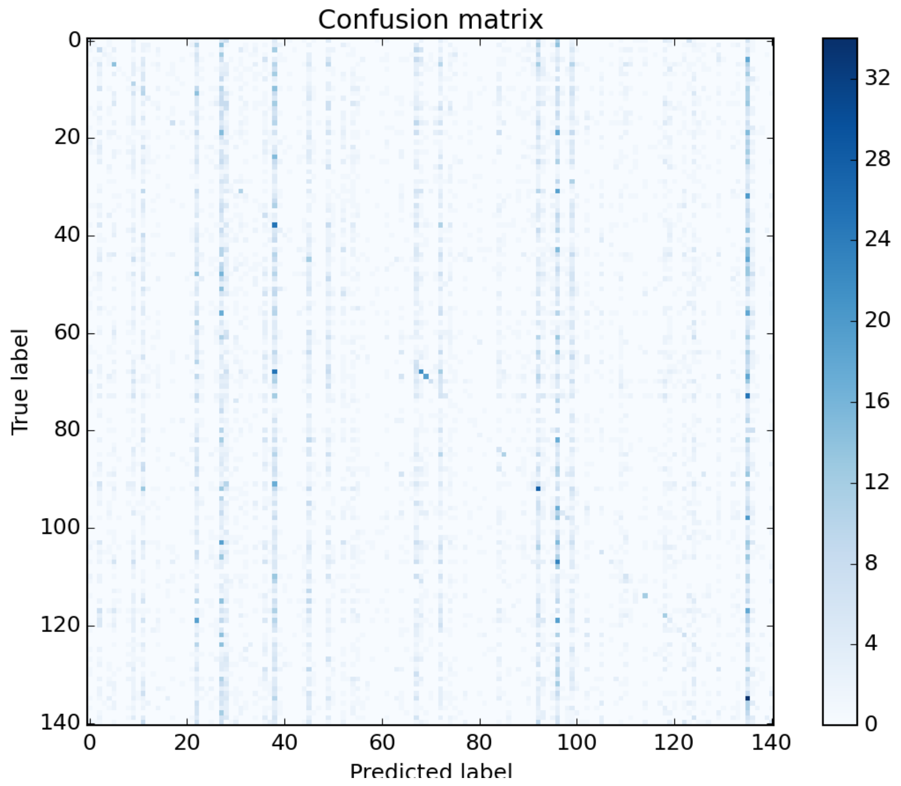

Figure 6: Confusion matrix for best model (18-word, 50 hidden layer size, lr 0.3)

7.2 Recurrent Neural Networks

For our problem, the subtleties present in the data and the large number of potential classifications proved a challenge to the RNN. With the short 18-word passages, our RNN models achieved a 0.095 test accuracy after 100 epochs. With the longer 50-word passages, the model performed approximately 2% better than the short-passage model at the same epoch.

From the training error plot, we can see that there is a slight overfitting present in all models. The model with a hidden-layer size of 100 demonstrated the highest discrepancy between train and test error, while not achieving as low a test error as the model with a hidden-layer size of 50. This seems to imply that after a certain hidden-layer size, the additional neurons do more to encourage overfitting than improving the model's actual performance.

Both short-passage and long-passage networks took several minutes to complete an epoch (on average 5.6 minutes for 18-word, 16.7 minutes for 50-word); given time-constraints and difficulty developing with our initial neural network implementation, we were unable to train either model for enough epochs to achieve convergence, and we were unable to train the 50-word passage models for 100 epochs. It's clear from the test graphs that training the larger hidden-size models would have resulted in better performance, as the velocity of the decreasing error rate is still high near 100 epochs.

7.3 Error Analysis

As shown by the confusion matrix in the previous section, the errors produced by the RNN model seemed to misclassify many authors as a single author (visualized by the vertical "stripes" in the plot). When authors are misclassified, there doesn't seem to be corresponding confusion for the other author, resulting in a one way error – many authors are classified as a single author, but works by this single author are rarely classified as belonging to one of the many authors. Interestingly, there appear to be several authors which the model predicts quite well; some of these authors coincide with vertical streaks. This may imply that there are a few authors with writing styles that are suitably generic, enough for the model to classify the works of many other authors into this single author. Some authors like Isaac Newton will have math and scientific language in his text, making his works much easier to classify.

One potential cause for error may be the use of pre-trained GloVe vectors. Because these word vectors are meant to capture similarities between words, certain words such as "therefore" and "furthermore" will have very similar word vectors. However, these similar word choices could determine an author's style. We can do additional analysis to see if over the course of training, word vectors of synonyms will become different.

8 Future Work

Increasing experiment duration Due to implementation difficulties with an earlier RNN implementation, we faced significant time constraints when training our models. Since our input data is very large, our RNN models require a significant amount of time to train a single epoch, which limited the number of epochs we could feasibly attempt. As a result, some of the higher hidden-layer-size or passage-length models have not had the chance to converge. Running the experiments for more epochs could allow these experiments to reach convergence, which would likely improve their performance on the test set significantly.

Additional experiments with hyperparameter tuning While we attempted some tuning of each model's hyperparameters by changing the hidden-layer size and learning rate, additional experiments with larger hidden-layer sizes, learning rates, and decay parameters could result in better performance.

Different network architectures We implemented our RNN models with a single hidden layer. Adding one or more additional hidden layers could allow the RNN to learn higher level features. Convolutional layers could enable the network to begin to recognize distinctive phrases or other subsequences of passages that might otherwise go unnoticed in more vanilla RNNs.

Dataset quality The dataset we used for our experiment was of high quality, but still contained internal metadata (e.g. table of contents, chapter headings, forewords) that may have been detrimental to our classification performance. Improving the dataset by removing these internal metadata and retaining just the raw text of works could reduce extraneous noise and misidentified passages, thereby increasing classification accuracy.

Other RNN Models There are many models that we can explore, that may be better than a vanila RNN. For example, we can train 2 RNNs, one for sentence level features that take in the word vectors of words as input, and one for paragraph level features that take in the ouputs of the sentence level RNN. However this method would only work on the 50 word input. Additionally, instead of having an output at each word of the sentence (and backpropagation), we could only have an output only at the end of sentences. This will significantly save computation time. Most words in the english language, such as 'the' or 'a', will not distinguish styles. We these words are fed into the

RNN, it will most likely not change the output at all. Additionally, there is the cold start problem as well. There is no use predicting the author over the first few words of the sentence, because it is not enough to distinguish between authors. It will only mislead the model and eat up additional computation.

9 Acknowledgements

We wish to thank the CS224D course staff for their assistance and guidance over the course of this project, and Shibamouli Lahiri from the University of Michigan for his high-quality Project Gutenberg dataset. We would also like to thank Microsoft Azure for providing compute credit for their G-series HPC clusters.

References

[1] Jose Nilo G Binongo and MWA Smith. The application of principal component analysis to stylometry. *Literary and Linguistic Computing*, 14(4):445–466, 1999.

[2] Shibamouli Lahiri. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. *ArXiv e-prints*, October 2013.

[3] Siham Ouamour and Halim Sayoud. Authorship attribution of ancient texts written by ten arabic travelers using a smo-svm classifier. In *Communications and Information Technology (ICCIT), 2012 International Conference on*, pages 44–47. IEEE, 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation.

[6] Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. Authorship attribution using probabilistic context-free grammars.

[7] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

[8] Fiona J Tweedie, Sameer Singh, and David I Holmes. Neural network applications in stylometry: The federalist papers. *Computers and the Humanities*, 30(1):1–10, 1996.