Authorship Detection and Authorship Style Detection in Classic Literature

Peter Zeng
Stony Brook University
pezeng@cs.stonybrook.edu

Abstract

Authorship detection is a task that has many applications and can be approached at many angles. While current work exists for the detection of plagiarism or for journal/article author detection, this project intends to apply this paradigm to classic literature and optimize the detection of authorship in novels. We start with LSTMs and GRUs, and then move on to bidirectional version of both as well. After that, we explore a more abstract method of trying to group different authors together utilizing K-Means and silhouette score to evaluate our clusters. [3]

1 Introduction

Literature plays an important role in nearly all aspects of human life, yet many people in this day and age either read very little for entertainment, or claim to have some favorite authors because of their *style*, and refuse to read anything else. What if we can devise a method of first detecting authors of literature, and from this see if these claims even make sense. Then, if so, how can we capture these similarities? This can be rather challenging of a problem because the first task involves long long corpora with a vast amount of training data required, and the second task remains relatively abstract.

For the first task, we want to utilize deep-learning recurrent neural networks that allow us to make use of sequence-to-sequence vector embeddings of our data. Since standard RNNs have the issue of vanishing gradients, we instead turn to RNNs with memory: LSTMs and their more efficient relative GRUs. Previous work has been done to apply these models on news datasets, specifically Reuter's 50-50 dataset [1]. Then, after tackling this issue, we take on the more abstract grouping of similar authors, which we will use clustering to attempt, as well as try to pull information from the best scoring clusters (scored using silhouette score) [2].

Since both of these tasks used different corpora, the methods are altered significantly. For processing book text, and to maintain authorship style, we decided to first find an optimal sequence length for our task of detection - elaborated on later, as well as preserving punctuation (as an attempt to preserve grammatical decisions by the author) before our other pre-processing tasks. This differs from the approaches in previous works as they attack their task from a pure tokenization problem and extract only the words, stripping all punctuation, and leaving little in the area of choices specific to the author.

The corpora used for this project was a set of raw texts obtained from the Stony Book Research team at Stony Brook University, which were extracted from the Gutenberg Project. We sampled a set of 100 books for further processing. The baseline was observed based on the LSTM, and standard tokenization was used, which were then passed into GloVe to get the final sequence embeddings to be used as input into the neural networks. The baseline results were basically a coin flip, as the max achieved results were around 0.5 for accuracy, and a 0.01 loss, with only a 0.33 F1 score. For loss, we utilized cross-entropy loss as this was a classification task. Specifically, the test involved processing the books until either a certain number of books were processed, or a certain number of authors was reached.

1. We implemented various recurrent neural networks for the task of authorship detection in classic literature.
2. We then cluster extracted sequences by dif-

ferent authors to observe similarities among authors.

3. Our evaluation for detection shows a significant improvement in every metric when utilizing GRUs and Bidirectional versions of both GRUs and LSTMs for the task of authorship classification (around 40-60 percent increase).

4. We were then able to extract some interesting observations from the best performing clusters (the clusters with the highest coherence).

2 Task

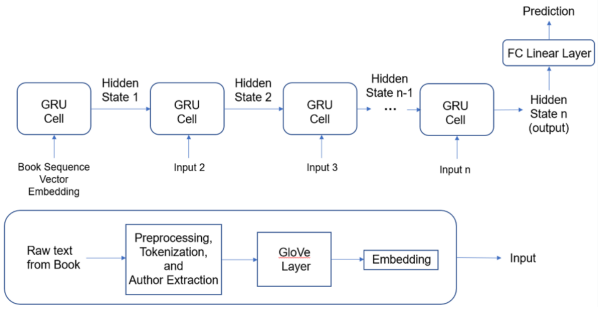
Our primary task is purely an authorship classifier, which takes in book text in the form of extracted sequences, and classifies these sequences as an author in a class of authors - a multi-class classification problem - utilizing deep-learning recurrent neural networks. The first challenge here was finding the optimal method of pre-processing the input data to be best suited for our task, which we accomplished by tokenizing the text while preserving punctuation. As for the optimal length, we wanted something that wouldn't be so short so that the important grammar isn't captured, but not so long as our models would have a tough time learning from these sequences. Then, the next challenge we faced was optimizing our neural networks, which involved tuning their hyper-parameters and then finding the best RNN with respect to all the evaluation metrics.

Our findings for the authorship classification task were that in terms of pre-processing, sequences of the longest length (100) words beat sequence embeddings of shorter lenghts. Further, the baseline was also beat by preserving grammar. Finally, the best RNN for this task turned out to be bidirectional GRU for all of accuracy, loss, and F1 score.

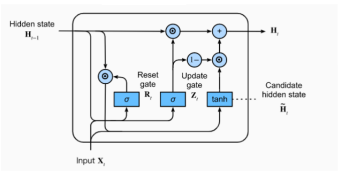
For the clustering task, we took separate embeddings and used batched K-Means clustering as a way to more efficiently achieve our end results, and evaluated our clusters using the clusters' silhouette scores. This measures the coherence of the clusters (how far each point in the cluster is away from the center of the cluster. Since there are no ground truths for author similarity, I maintained the parameters from authorship classification for the task of author grouping.

2.1 Baseline Model

Pictured below is the baseline GRU model used for classification.



The baseline model first creates a GloVe embedding from the raw text to use as input into the GRU. A GRU, or a Gated Recurrent Unit, is a sequential neural network with *gates* which allow the neural network to keep a form of memory of previous inputs. Below is one GRU cell and its associated functions:



$$z = \sigma(W_z \cdot x_t + U_z \cdot h_{(t-1)} + b_z)$$
$$r = \sigma(W_r \cdot x_t + U_r \cdot h_{(t-1)} + b_r)$$
$$\tilde{h} = tanh(W_h \cdot x_t + r * U_h \cdot h_{(t-1)} + b_z)$$
$$h = z * h_{(t-1)} + (1 - z) * \tilde{h}$$

These functions include the calculations for the current cell state, update and reset gate, and a candidate function. As such, after we input our embeddings into the GRU, we can use it to predict the label in the final fully connected layer, thus accomplishing our multi-class classification task. Once this is accomplished and we have our optimal parameters, we can move on to the clustering task, which utilizes Word2Vec embeddings, and then creates the clusters using mini-batch K-Means. The most important part of this process are the calculated silhouette scores, which will be elaborated shortly.

2.2 Issues

While pictured is actually the GRU model, the issues lie within the LSTM baseline, for a number of reasons. The main reason is that shortly put, GRUs are able to handle shorter language sequences more efficiently and with better efficacy. For this task, since each sequence is relatively shorter (grammar and phrases are what we are trying to capture in our training), LSTMs perform much worse. Furthermore, in terms of *computational performance*, LSTM also loses to GRUs as

LSTMs are inherently the more complex network architecture. Please note that this doesn't mean that longer sequences automatically indicate that LSTM would perform better as there is a certain point where that metric will fall off, thus we aim to optimize this with our experiments in section **3**.

3 Experiments

Thus, our approach is to then test the performance of additional RNNs, as well as try differences in data pre-processing. In particular, we wanted to optimize the length of the sequences to be embedded, as well as try different modifications to our text before tokenizing and embedding.

3.1 Sequence Length Optimization on GRU and LSTM:

For the pre-processing, when extracting sequences from the raw book text, we will set the sequences to lengths of *25, 50, 100* and see which sequence length provides the best results in evaluation.

3.2 GRUs vs LSTMs vs BiGRUs vs BiLSTMs

We will additionally try bidirectional version of our two main networks, and see which one provides the best results on the length we found to be optimal.

3.3 Does Grammar *Really* Matter?

We will see if our proposed hypothesis of removing punctuation from our words before tokenizing actually makes a difference.

3.4 Clustering Task to See Who Similar Authors Might Be:

We will test our clustering algorithm and calculate the silhouette score of our clusters to first evaluate if there are any good clusters, and if so, pull the top ranking sequences in the cluster to see which authors they were.

4 Evaluation

4.1 Dataset Details

The original corpora consists of the Gutenberg Project's library of classical literature titles, which was compiled by the Stony Book Research team at Stony Brook University. Specifically, I used the raw texts pulled from Gutenberg, from which I stripped all unnecessary info such as the Index, Glossary, various chapter titles. Then, I extracted the author as the label for the data. Finally, I split the text into sequences of variable length, of which we're optimizing. Once the optimal parameters are found for setting up our data, we'll use that format for our clustering task.

4.2 Evaluation Measures

For our classification task, we used three different metrics for evaluation: accuracy, loss, and F1 score. Specifically, we will be using Cross-Entropy Loss, because our task is *multi-class classification* which means we'll be calculating the loss among *vectors* [4]. F1 score is another way of evaluating accuracy, which is derived from the precision and recall of the test, as a way of comparing the two.

$$precision = \frac{\text{\# of True Positives}}{\text{Total \# of Positives}}$$

$$recall = \frac{\text{\# of True Positives}}{\text{\# of True Positives + \# of False Negatives}}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

For the clustering task, we will utilize the silhouette score of the clusters, which is a metric derived from the mean distance of the points in a cluster to its center, as well as each individual point in the cluster.

4.3 Implementation Details

In terms of the hyper-parameters, we wanted a learning rate that would minimize the loss in every network, and as for the number of epochs, we wanted a number of epochs that didn't overfit on the training data. For fine-tuning of these two parameters, we tried various learning rates, as well as different numbers of epochs to see which would perform the best. What resulted was the baseline hyper-parameters were already around optimal, so we kept them as they were: Learning Rate: 1e-2, Epochs: 20.

4.4 Results

Finally, we have the results of our various experiments:

Sequence Length

We are only showing accuracy here, although all metrics were taken into account when choosing the best model. Furthermore, despite the other metrics not being shown, they were in line with the best accuracy. One final note is that the training data is built on a sample of books from the 100

books (10 books), to allow for more efficient computation.

	25	50	100
GRU Accuracy	0.64	0. 8	0.837
LSTM Accuracy	0.166	0.16	0.16

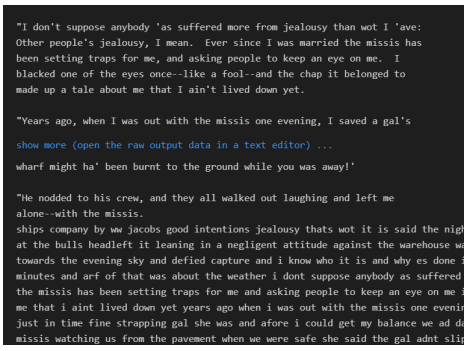
GRU vs LSTM vs BiGRU vs BiLSTM

We show all metrics to determine which RNN is the best for the task of authorship classification in classical literature. We maintain the optimal length we found in the previous experiment (100 words).

	GRU	LSTM	BiGRU	BiLSTM
Accuracy	0.797	0.15	0.863	0.64
Loss	0.011	0.033	0.0076	0.0155
F1 score	0.681	0.026	0.803	0.435

Punctuation or Not

We show the effects of having punctuation, capitalization, etc or not in the pre-processing step, and we use our optimal model (BiGRU) found in the previous experiment. For the punctuation removed, we specifically lowercase all words, remove appended text, remove multiple spaces, remove ellipsis, replace dashes, and remove all other punctuation. Pictured is a visual representation of this change.



	Accuracy	Loss	F1 Score
Punctuation	0.863	0.0076	0.803
No Punctuation	0.462	0.023	0.284

Clustering

For this task, we used Word2Vec embeddings of length 100, and used mini-batch k-means to cluster the embeddings. We then calculated the Silhouette score of each cluster, and got some fairly respectable scores. For a base task of embeddings by 5 authors, and 3 clusters, we got a max average Silhouette score of **0.34**, and a max Silhouette score in that cluster of **0.52**. The top 3 embeddings in this cluster were by Angela Brazil, Joseph Conrad, and Angela Brazil, respectively. Then, for another bigger experiment with embeddings by 10 authors, and 3 clusters again, we got a max average Silhouette score of **0.31** and a max Silhouette score in that cluster of **0.49**. The top 3 embeddings in that cluster were by Joseph Conrad, L. FrankBaum, and Henry James, respectively.

4.5 Analysis

1. The BiGRU significantly outperformed all other neural networks, with the regular GRU trailing somewhat behind. The LSTM performed by far the worst in the classification task.

2. Removing punctuation and utilizing pure tokenization is **far** worse than keeping it for both tasks. This makes sense as an author's style is much more than just the words they choose to use, and as such preserving the punctuation is best for both tasks, and definitely shown in the classification task.

3. The best sequence length that we found was a length of 100. However, that doesn't necessarily mean it's the optimal length. Since testing this would take some time, we don't really know if a sequence length of 100 is better than all other lengths, but we do know that it is better than the other values we tried.

4. The clustering task was rather abstract, and as such, there might've been a few places for error. Furthermore, because our Silhouette score average **was** so low, it means we can't really be that confident in our results listed prior. In order to test this, we would have to create a ground truth, perhaps from annotators who are well-read in classical literature, and use that as a baseline for comparing our clustered results to.

5. For the entire project, we made the rather naive (but unavoidable) decision to only sample 100 books from the Gutenberg Project Library. This wasn't done as a choice, but rather out of necessity. However, because we chose to do so, we might not be able to replicate all these results using other samples of the same corpora. Simply put, these experiments aren't entirely deterministic.

4.6 Code

Link to code and data on Google Drive!

5 Conclusions

Overall, we had some very successful experiments. We optimized the pre-processing task and found a good length for sequences, and then found out that the best model for the classification task was BiGRU. Furthermore, we confirmed our hypothesis that preserving punctuation would be better than removing punctuation for our classification task. Not only did we confirm that hypothesis, however, but the results for preserving punctuation were **far** better than the results with no punctuation.

For the clustering task, we had originally not had much hope for the quality of the clustering. The baseline performance initially produced results of around 0.2 for the average Silhouette score, which is very very bad. However, after optimizing for the parameters in our classification task, we believe that we improved our clustering quality by a significant amount. That being said, we didn't create the metric that was originally proposed, and that leads us to future work that we would like to implement.

5.1 Future Work

First, for both tasks, we would like to use some more advanced transformer based embeddings and use those for the classification task (BERT for example). Furthermore, in terms of actually creating a *metric* for the purpose of finding similar styles between different authors, we thought of a way to combine somehow (average as a naive start) the sequence embeddings of each author and use that as a starting point for the creation of said metric. However, we didn't have enough time to flesh this concept out. In order to get a ground truth for this, we would ask survey participants to rank or group authors, at least to get a starting point, in order to better evaluate our clustering task.

6 References

[1] Narayanan, Ranjani. (2020). Deep Learning based Authorship Identification.
[2] How to Cluster Documents using Word2Vec and K-Means
[3] An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means Clustering Validity
[4] Cross-entropy for classification