Towards a Universal Document Encoder for Authorship Attribution

Kieran Sagar Parikh
Department of Computer Science
Middlebury College
Middlebury, Vermont, USA
kparikh@middlebury.edu

Vinodini Venkataram and Jugal Kalita
Department of Computer Science
University of Colorado
Colorado Springs, Colorado, USA
vvenkata@uccs.edu, jkalita@uccs.edu

Abstract

Distributed embeddings of words and sentences have been successful in representing the semantic and syntactic properties of text. In this paper, we examine the creation of distributed document embeddings based on textual style. We present a supervised training architecture to train an encoder to produce these embeddings. We explore the viability of these embeddings for the authorship attribution task, and assess the effects of encoder architecture, text processing, and classifier architecture on authorship attribution performance. While our model does not meet or exceed state of the art results on the same datasets, we are able to confirm our hypothesis that our supervised encoder training method produces an encoder which embeds texts based on that which unites an author's own work and distinguishes it from others, collectively speaking.

Introduction

Writing style represents the distinguishing characteristics of a given author’s writing, and characterizing writing style is a central topic in literary and forensic scholarship. Writing style plays a crucial role in authorship analysis tasks.

In this paper, we present the development of a text encoder which learns to produce document vectors reflecting the author-specific (stylistic) qualities of texts. We follow by exploring the viability of such an encoder for authorship attribution. Authorship attribution is the process of inferring characteristics of multiples authors’ writing styles from examples of their work, and subsequently using these inferred characteristics to classify unseen texts by author (Juola, 2008).

Distributed representations or so-called embeddings are widely used to represent the semantic properties of words, sentences, and snippets of text (Mikolov et al., 2013; Kiros et al., 2015; Le and Mikolov, 2014). Embeddings have also been shown to be useful in a limited way in capturing the stylistic qualities of an author for use in authorship attribution (Koppel, Schler, and Argamon, 2011; Gomez-Adorno et al., 2018). Most such embeddings have been computed using unsupervised approaches, based on the context in which a word or sentence appears. However, Conneau et al. (2017) have recently taken a supervised approach to teach an encoder to create “universal” sentence embeddings.

Our research focuses on evaluating the performance of an encoder-classifier model for authorship attribution. We take a supervised learning approach to create an encoder to effectively map texts to embeddings based on (the stylistic) characteristics of their writing. We use these embeddings with various classification architectures to perform experiments in authorship attribution. We believe that our research could be extended to create a universal encoder to create (style) embeddings for any texts for later use in classification based on the encoded (stylistic) properties.

An encoder-classifier model has several advantages over an end to end model for author attribution. First, embeddings produced by such an encoder may be useful in a variety of authorship analysis tasks, such as classifying texts based on age or gender, or clustering authors based on stylistic similarities. Second, an encoder that maps texts with different styles to distant vectors in a vector space could also help in open-set author attribution to identify writings of unknown authors. An open-set classifier could use a threshold maximum distance in the vector space to detect writing samples of unknown authors. Third, by breaking up the model into two distinct pieces (encoder and classifier) end-user flexibility is increased, enabling, for instance, encoding to happen on a mobile device and classification to happen at a later point, perhaps on a centralized server. Furthermore, such an architecture has the added benefit that different classification algorithms can be used on the same embedding without having to re-encode the relevant text.

Finally, in the universal case, the encoder needs to be trained only once to develop a general ability to encode texts, but then could be used to solve different authorship attribution problems by training only a classifier on the relevant dataset after it has been encoded. This could significantly decrease the time and resources required to solve an authorship attribution problem.

Related Work

Koppel, Schler, and Argamon (2011) represented each text as a vector based on the frequencies of 250,000 unique, but overlapping, space free character 4-grams. A new text is assigned to the author whose texts are closest to it in terms of cosine similarity in the vector space. The authors addressed the open-set version of the problem, where an anonymous text that is not similar enough to any known texts is not assigned to any candidate author.
More recently, the use of word embeddings computed using Word2Vec (Mikolov et al., 2013) or GloVe (Pennington, Socher, and Manning, 2014) in an unsupervised manner have become commonplace. Word2Vec was further extended with the introduction of paragraph vectors, which are learned embeddings of variable-length texts (Le and Mikolov, 2014). Paragraph vectors are learned at the same time as the word embeddings to encode the meanings of all words in the paragraph, thereby preserving an understanding of the ordering of words within a text.

Kiros et al. (2015) presented an unsupervised approach to train an encoder to produce generic distributed sentence encodings. Adapting the skip-gram model of Mikolov et al. (2013) to sentences, they used a new objective function to encode a sentence based on others around it. The chosen to use gated recurrent units (GRUs) to both encode a sentence and decode the previous and next sentence from such an embedding.

Conneau et al. (2017) created an encoder to produce universal sentence embeddings using a supervised approach. They used a dataset with 570 thousand sentence pairs labelled as having one of three semantic relationships: entailment, contradiction or neutral. The training model created separate embeddings for the two sentences in a pair. Three matching methods were applied to these embeddings and the result was then fed into a 3-class classifier. The quality of the learned embeddings were evaluated by their performance in transfer tasks, such as sentiment classification, semantic relatedness, paraphrase detection and semantic textual similarity. They found that a BiLSTM network with max pooling produced the best embeddings, outperforming SkipThought vectors and requiring less training time.

Sari, Vlachos, and Stevenson (2017) adapted the FASText method to perform text classification (Joulin et al., 2017) for authorship attribution. They learned custom word and character n-gram embeddings at classifier training time. The embedding for a text was found by averaging the embeddings of all its n-grams, and it was fed to a linear classifier. They achieved state-of-the-art classification accuracy on benchmark datasets.

Gomez-Adorno et al. (2018) use paragraph vectors for authorship attribution. They learn paragraph embeddings considering character, word, and POS n-grams, inspired by the knowledge that character n-grams have had considerable success in authorship attribution. The embeddings are used with a logistic regression classifier to perform cross-topic authorship attribution. They find that the best model is embeddings created from the concatenation of vectors trained on POS 1, 2, 3-gra ms and Word 1, 2, 3-gra ms.

Architecture and Experimental Protocol

There are two main steps in the way we go about authorship attribution. First, we train a Siamese twin network by providing it with pairs of documents as input. Once the encoder has been trained using this architecture, we encode all our documents using the learned encoder weights. In the second step, we use a decoupled machine learning technique to classify the the documents based on their embeddings.

Siamese Twin Encoder

We adapt the supervised learning approach for sentence embeddings used by Conneau et al. (2017) to develop an encoder to create embeddings based on authorship style. We modify their training model to take in two variable length snippets of text with a binary label indicating whether the two snippets were written by the same author.

The Siamese encoder training architecture we use is conceptually illustrated in Figure 1. Siamese networks, where two identical sub-networks are used to train on a pair of inputs at the same time have been used successfully for signature verification (Baldi and Chauvin, 1993; Bromley et al., 1994), face verification (Hu, Lu, and Tan, 2014) and other image recognition tasks (Koch, Zemel, and Salakhutdinov, 2015) that require learning suitable distance metrics, adapted to a dataset or domain. However, in practice, we have only one encoder. During training, the encoder is first used to obtain an embedding \( u \) of the first text of the pair. The same encoder is next used to obtain the embedding \( v \) of the second text of the pair. Once both embeddings are in place, three methods are applied to them: concatenation, element-wise product, and absolute element-wise difference, to produce a vector that combines the embedded information for the two documents. This resulting vector is passed to a dense neural network, consisting of two 64-node hidden layers and a softmax layer, which performs binary classification. Backpropagation on the weights of the encoder is performed based on the errors in pairwise classification, i.e., if the pair was judged correctly to be authored by the same or different authors, as in the input.

We hypothesize that this approach will force the encoder to learn to make embeddings based on that which unites an author’s own work and distinguishes it from others’, collectively speaking, ultimately resulting in embeddings of documents that are grouped by author within the vector space.

Generating Training Pairs for Encoder

The datasets used in this research contain individual texts labelled by author. It would be computationally expensive...
to attempt to train on the complete set of all possible pairs given its large size\textsuperscript{1}, and would also likely result in significant overfitting as the number of training pairs would be much larger than the number of unique training texts. Therefore, pairs of texts are created at the start of each training epoch. In the first training epoch, 100 differing author pairs and 100 same author pairs are created for each author in the dataset. Differing author pairs are created by holding constant, the author of the first text and randomly choosing a second author for each pair. Same author pairs are created by holding constant the author of both pieces of text in a pair. For every pair, texts are chosen randomly from among the relevant author’s works. In subsequent epochs, pairs are created both randomly and adaptively based on the author pairs on which the model performed poorly in the previous epoch. Ten differing author pairs and fifty same author pairs are created for each author in the dataset. A misclassification count from the previous epoch organized by author pair is used to create ten training pairs for each misclassified pair, with a maximum of hundred training pairs for any unique author pair. Pairs created based on previous misclassifications are created using the same authors as the misclassified pair, with texts chosen randomly from among the relevant authors’ works. Pair generation is done in this way to avoid overfitting to specific pairs in the set of all possible pairs, and to adaptively focus training on difficult author pairs.

**Encoder Models Used**

We investigated variation of both the fundamental architecture of the encoder and the way in which an input text is prepared for the encoder. We investigated four encoder architectures: a bidirectional LSTM (BiLSTM) with max pooling, a BiLSTM with mean pooling, a Hierarchical Convolution Neural Network (ConvNet) with max pooling, and a ConvNet with mean pooling.

In terms of preparation, three main encoder models were used. The first model used was a word based model which fed texts encoded using word embeddings into the LSTM to create 256 dimensional embeddings. The second was a character based 3-gram model which encoded texts using character 3-gram embeddings and used the LSTM to create 256 dimensional embeddings. The third model was a combination of the first two, using two encoders in parallel, one operating on words and the other on character 3-grams, to produce 512 dimensional embeddings.

Furthermore, Sari, Vlachos, and Stevenson (2017) found word and character n-gram embeddings learned directly on the dataset to be effective in authorship attribution, so our models were tested using both pretrained GloVe word embeddings (Pennington, Socher, and Manning, 2014) and pre-trained character 3-gram embeddings from Hashimoto et al. (2016) as well as custom word and character embeddings. Custom word and character 3-gram embeddings were produced using the skip-gram model (Mikolov et al., 2013) on both the train set exclusively and the combined train and test set of the CCAT-50 dataset.

\textsuperscript{1}For instance, over 3 million unique pairs could be created from the 2500 training set examples in the CCAT-50 dataset.

**Metric Learning**

Since our model aims to produce document embeddings such that documents are grouped by author within the vector space, the effect of using metric learning techniques to group classes together and separate them from other classes was investigated. We used Large Margin Nearest Neighbor (LMNN) (Weinberger and Saul, 2009) metric learning, which is a metric learning algorithm designed specifically to improve \(k\)NN performance. LMNN learns a Mahalanobis distance metric in a supervised way where, for a given data point, it is rewarded if the point’s \(k\) nearest neighbors (measured using the learned metric) are of the same class as the data point, and is penalized if any of its \(k\) neighbors are of a different class.

**Classification Methods**

As mentioned earlier, the Siamese twin network has a single encoder in practice, although it is illustrated as if there are two parallel encoders. The purpose of training the Siamese network is to train this encoder to produce document embeddings that can discriminate among authors. Thus, after training, the encoder is used to produce embeddings of all texts in both the training and test sets. These embeddings are then fed to various decoupled classifiers for the authorship attribution task. All classifiers are trained on the embeddings of the training set and tested on the embeddings of the test set. The first classification algorithm uses SVMs with RBF kernels in a “one-against-one” (Knerr, Personnaz, and Dreyfus, 1990) approach to perform multi-class classification. The second algorithm is \(k\)-nearest neighbors (\(k\)NN) with a \(k\) value of 5. The SVM and \(k\)NN classifications are performed using Scikit-learn (Pedregosa et al., 2011). The third classifier is a cohort algorithm which takes advantage of the binary classifier learned during encoder training. Each unknown text is compared to 30 training set texts from each author using the binary classifier, which predicts whether or not two texts were written by the same author. When the binary classifier predicts that two texts were written by the same author, it is counted as a vote for author of the text to which the unknown text is being compared. Votes are tallied for each author and the text is attributed to the author with the most votes.

Various methods of ensembling classifiers were also investigated. Four basic architecture were tested.

**Discrete Plurality Voting** The first is a discrete plurality voting ensemble. Four classifiers are used in the ensemble: \(k\)NN, SVMs with RBF kernels in a “one-against-one” (Knerr, Personnaz, and Dreyfus, 1990) approach, the cohort algorithm described earlier, and a dense neural network with two 256 node layers, one 64 node layer, and a softmax layer, trained for 500 epochs. Each classifier votes for one class and all votes are weighted equally. Ties are broken by choosing \(k\)NN.

**Meta-Classifier** The second ensemble uses a meta-classifier, which seeks to predict which classifier(s) (\(k\)NN, SVM, etc.) will correctly predict the author of an example given its embedding. The same four classifiers from the discrete plurality voting ensemble are used. Predictions for the
training set from each classifier are used to label each embedding in the training set with the classifier(s) that correctly predicted it. The meta-classifier is trained using the training set embeddings and these labels, and is subsequently used on the test set embeddings to predict which classifiers will correctly predict this example. For each various different meta-classifiers were tested, including SVMs, decision trees, naive Bayes, logistic regression, and XGboost (Chen and Guerstrin, 2016).

**XGBoost Logits** The third ensemble architecture uses XGBoost (Chen and Guerstrin, 2016) to predict an example’s class given the raw logits from each classifier. The same four classifiers from the discrete plurality voting ensemble and meta-classifier ensemble are used. XGBoost is trained using the raw logits and class label for each example in the training set. It is used with the raw logits from the test set examples to predict their classes.

**Soft Voting** The fourth ensemble architecture uses soft voting, which predicts an example’s class using the max of the sums of the probabilities for each class from each classifier. Four classifiers are used in the ensemble: kNN, SVMs with RBF kernels in a “one-versus-rest” approach, Random Forest, and Logistic Regression.

**Experiments, Results, & Analysis**

This section describes the datasets used and the experiments performed along with the results obtained.

**Datasets Used**

Three datasets were used to evaluate the viability of this approach for authorship attribution.

**CCAT-50** (Houvardas and Stamatatos, 2006) is a set of 5000 news stories, comprised of 50 training texts and 50 test texts for each of 50 authors. All texts are corporate news stories to reduce the effect of genre on classification. This dataset is a subset of the Reuters Corpus Volume 1.

**CCAT-10** (Stamatatos, 2008) is a subset of the CCAT-50 dataset with fewer authors. It contains 50 training texts and 50 test texts for 10 authors. Like CCAT-50, all texts are corporate news stories to reduce the effect of genre on classification.

**IMDb62** (Seroussi, Zukerman, and Bohnert, 2011) is a set of 62,000 movie reviews from the Internet Movie Database (IMDb). It contains 1,000 texts from each of 62 authors. This dataset was split into a train and test set by setting aside 10% of each author’s texts for the test set and using the remaining 90% as the training set.

<table>
<thead>
<tr>
<th>Model</th>
<th>CCAT-50</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>62.9</td>
<td>63.8</td>
<td>61.36</td>
<td>75.4</td>
<td>76.6</td>
<td>68.2</td>
<td>89.9</td>
<td>88.1</td>
<td>78.1</td>
<td></td>
</tr>
<tr>
<td>Char 3-grams</td>
<td>48.2</td>
<td>49.3</td>
<td>40.4</td>
<td>66.8</td>
<td>67.2</td>
<td>66.0</td>
<td>47.7</td>
<td>35.1</td>
<td>28.7</td>
<td></td>
</tr>
<tr>
<td>Combined</td>
<td>59.6</td>
<td>61.8</td>
<td>55.7</td>
<td>72.6</td>
<td>73.0</td>
<td>67.0</td>
<td>88.5</td>
<td>84.7</td>
<td>71.6</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Multiclass classification accuracy (%) by encoder architecture, classifier type, and dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>CCAT-50</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
</tr>
</thead>
<tbody>
<tr>
<td>kNN</td>
<td>63.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kNN w/ LMNN</td>
<td>62.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Multiclass classification accuracy (%) for word-based model using kNN classifier

**Preliminary Results**

To obtain preliminary results, we trained biLSTM with max pooling encoders on word, character 3-gram, and combined models for all three datasets. We used the trained encoders to produce embeddings for each dataset which were subsequently fed to SVM, kNN, and Cohort classifiers. Results are presented in Table 1. Our word-based model achieved higher classification accuracy than the other two models on all datasets.

We also observed that kNN and SVM classifiers had less than 2% difference in classification accuracy on all three datasets using the word model, and on CCAT-50 and CCAT-10 using the character 3-gram and combined models. Nevertheless, all models and classifiers fail to meet state of the art results on these datasets. A comparison of this research against the state of the art is provided in Table 3.

As a result, further experimentation was undertaken to both attempt to improve classification accuracy to state of the art levels and assess the advantages of this approach. In the subsections that follow, the effects of metric learning, variation of encoder architecture, the use of custom word and n-gram embeddings, and the use of ensemble-ized classifiers are presented. Finally, a visual representation of the documents in the CCAT-10 dataset is presented and analyzed.

**Metric Learning**

Given the success of kNN classifiers, the effect of applying the Large Margin Nearest Neighbor (LMNN) (Weinberger and Saul, 2009) metric learning algorithm to the embeddings was investigated with a k value of 5. Results from these experiments are presented in Table 2. It was found that the use of LMNN caused a small decrease in classification accuracy on both the CCAT-50 and and CCAT-10 datasets.

**Custom Embeddings**

Custom word and character 3-gram embeddings were produced using the skip-gram model (Mikolov et al., 2013) on both the train set and the combined train and test set of the CCAT-50 dataset. These embeddings were then used to train biLSTM encoders and perform classification. Results from these experiments are presented in Table 4.
Table 3: Comparison against other reported results.

<table>
<thead>
<tr>
<th>Model</th>
<th>CCAT-50</th>
<th>CCAT-10</th>
<th>IMDb62</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous n-gram words (Sari, Vlachos, and Stevenson, 2017)</td>
<td>70.16</td>
<td>77.80</td>
<td>87.87</td>
</tr>
<tr>
<td>Continuous n-gram char (Sari, Vlachos, and Stevenson, 2017)</td>
<td>72.60</td>
<td>74.80</td>
<td>94.80</td>
</tr>
<tr>
<td>SVM with affix &amp; punctuation 3-grams (Sapkota et al., 2015)</td>
<td>69.30</td>
<td>78.80</td>
<td>-</td>
</tr>
<tr>
<td>D2V words (Posadas-Duran et al., 2017)</td>
<td>71.84</td>
<td>80.80</td>
<td>-</td>
</tr>
<tr>
<td>D2V words + 2 + 3-grams (Posadas-Duran et al., 2017)</td>
<td>75.24</td>
<td>82.80</td>
<td>-</td>
</tr>
<tr>
<td>D2V words + 2 + 3 + 4 + 5-grams (Posadas-Duran et al., 2017)</td>
<td>74.84</td>
<td>84.60</td>
<td>-</td>
</tr>
<tr>
<td>SVM with bag of local histograms (Escalante, Solorio, and Montes-y Gmez, 2011)</td>
<td>-</td>
<td>86.40</td>
<td>-</td>
</tr>
<tr>
<td>Token SVM (Seroussi, Zukerman, and Bohnert, 2013)</td>
<td>-</td>
<td>-</td>
<td>92.52</td>
</tr>
<tr>
<td>Words - Discrete Plurality Voting Ensemble</td>
<td>64.80</td>
<td>76.20</td>
<td>89.56</td>
</tr>
<tr>
<td>Words - Soft Logit Voting Ensemble</td>
<td>65.64</td>
<td>77.00</td>
<td>91.69</td>
</tr>
</tbody>
</table>

Table 4: Multiclass classification accuracy (%) by embedding type.

<table>
<thead>
<tr>
<th>Model</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words (pre-trained)</td>
<td>62.9</td>
<td>63.8</td>
<td>61.4</td>
</tr>
<tr>
<td>Words (train)</td>
<td>47.8</td>
<td>49.6</td>
<td>39.5</td>
</tr>
<tr>
<td>Words (train+test)</td>
<td>52.6</td>
<td>52.1</td>
<td>44.7</td>
</tr>
<tr>
<td>Char 3-grams (pre-trained)</td>
<td>48.2</td>
<td>49.3</td>
<td>40.4</td>
</tr>
<tr>
<td>Char 3-grams (train)</td>
<td>49.5</td>
<td>50.6</td>
<td>51.8</td>
</tr>
<tr>
<td>Char 3-grams (train+test)</td>
<td>50.3</td>
<td>50.4</td>
<td>52.6</td>
</tr>
</tbody>
</table>

Interestingly, it was found that custom embeddings increased the accuracy of the character based model but decreased the accuracy of the word based model. This indicates that the semantic nature of pretrained GloVe embeddings may be important in the production of meaningful text embeddings, since the custom word embeddings probably captured less semantic value than the GloVe embeddings given the much larger training set used to create the GloVe embeddings. This is further evidenced by the fact that the char n-grams, which do not have any semantic value, performed better with custom embeddings. If this is the case, this would also explain the word model’s better performance on IMDb62 compared with CCAT-10 and CCAT-50, as previous users of these datasets have noted the importance of text topic for discrimination of authors in IMDb62 (Sari, Vlachos, and Stevenson, 2017; Seroussi, Zukerman, and Bohnert, 2013), while it is less influential in CCAT-10 and CCAT-50 as these datasets have already been controlled for topic.

Unsurprisingly, accuracy with both words and characters was slightly higher using custom embeddings trained on the combined train and test set when compared with custom embeddings trained only on the train set. This gain in accuracy was larger for the word model, likely because there are more words in the test set that are not in the train set than there are character 3-grams that are only in the test set.

Encoder Type

We varied the type of encoder that was used to observe its effect on classification accuracy. Results from these experiments are presented in Table 5. We found that no new encoder architecture could outperform our previous best classification accuracy of 63.8%, obtained using a biLSTM with max pooling and a kNN classifier.

Ensemble Classifier

Analysis of the test examples misclassified by the various classifiers revealed that the set of correctly classified examples from each classifier do not completely overlap. For the standard word-based biLSTM with max pooling with the CCAT-50 dataset, it was found that 582 examples—or 23.3% of the dataset—were misclassified by all classifiers. This shows that ensemble-izing these classifiers could result in a higher classification accuracy than the current best of 63.8% using only kNN.

The results of these experiments are presented in Table 6,

Table 5: Multiclass classification accuracy (%) by encoder type.

<table>
<thead>
<tr>
<th>Encoder Architecture</th>
<th>SVM</th>
<th>kNN</th>
<th>Cohort</th>
</tr>
</thead>
<tbody>
<tr>
<td>biLSTM w/ max pooling</td>
<td>62.9</td>
<td>63.8</td>
<td>61.4</td>
</tr>
<tr>
<td>biLSTM w/ mean pooling</td>
<td>58.4</td>
<td>61.8</td>
<td>61.6</td>
</tr>
<tr>
<td>ConvNet w/ max pooling</td>
<td>52.2</td>
<td>52.0</td>
<td>50.6</td>
</tr>
<tr>
<td>ConvNet w/ mean pooling</td>
<td>35.3</td>
<td>56.8</td>
<td>53.8</td>
</tr>
</tbody>
</table>

Table 6: Multiclass classification accuracy (%) for word-based model using ensemble-ized classifiers

<table>
<thead>
<tr>
<th>Ensemble Architecture</th>
<th>CCAT-50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Plurality Voting</td>
<td>64.8</td>
</tr>
<tr>
<td>Meta-Classifier</td>
<td>SVM</td>
</tr>
<tr>
<td>Meta-Classifier</td>
<td>Decision Tree</td>
</tr>
<tr>
<td>Meta-Classifier</td>
<td>Naive Bayes</td>
</tr>
<tr>
<td>Meta-Classifier</td>
<td>Logistic Regression</td>
</tr>
<tr>
<td>Meta-Classifier</td>
<td>XGBoost</td>
</tr>
<tr>
<td>XGBoost Logits</td>
<td>max depth = 6</td>
</tr>
<tr>
<td>XGBoost Logits</td>
<td>max depth = 20</td>
</tr>
<tr>
<td>XGBoost Logits</td>
<td>max depth = 50</td>
</tr>
<tr>
<td>XGBoost Logits</td>
<td>max depth = 100</td>
</tr>
<tr>
<td>XGBoost Logits</td>
<td>max depth = 256</td>
</tr>
<tr>
<td>Soft Logit Voting Ensemble</td>
<td>65.6</td>
</tr>
</tbody>
</table>
and a comparison against state of the art results is presented in Table 3. It was ultimately found that the soft voting ensemble achieved the highest classification accuracy, but still failed to meet state of the art results on all three datasets.

**Visual Analysis of CCAT-10 Documents**

The document embeddings produced by our word-based biLSTM with max pooling encoder for the CCAT-10 dataset were used to produce visual representations of the distribution of the encoded texts in the vector space. The vectors were reduced from 256 to 2 dimensions using t-SNE (Maaten and Hinton, 2008). The training set is represented in Figure 2, while the test set is represented in Figure 3.

Comparative analysis of the two figures can help to explain some of our results. Examples from the train set appear to be clustered by class, with visible margins between class clusters, few outliers, and little class overlap. Test set examples, however, appear less tightly clustered and have significant class overlap for some classes. While each author class has its own unique cluster in the train set, only four authors (1, 2, 4, and 7) have their own unique cluster in the test set. Two large clusters contain examples from the remaining classes: one with authors 5, 6, and 9, and another with authors 0, 3, and 8.

The tighter clustering in the train set when compared with the test set explains the performance of kNN on this dataset. During kNN, while the train examples to which a test example is compared may be clustered closely together by class, the test examples tend to be less tightly clustered and have more variance in position. Some may, therefore, be closer to a different class’s cluster in the train set, resulting in test example misclassification.

This also explains the poor performance of metric learning techniques: examples in the training set were already clustered by class with margins between clusters, so it is unlikely that any metric could be learned from this train data that would bring closer the test examples which are far from their classes’ clusters in the train set.

Despite the tighter clustering of train set examples when compared with test set examples, the encoder still manages to embed unseen examples to the same general area as the train examples of the same class, even if it does not manage to embed them precisely enough to achieve better authorship attribution performance. For instance, the large overlapping cluster in the test set containing authors 5, 6, and 9 occupies roughly the same space as the three individual clusters for authors 5, 6, and 9, in the train set. A similar phenomenon can also be observed with the other overlapping cluster in the test set. This indicates that the encoder is learning to embed documents based on the stylistic characteristics that distinguish the authors in a way that does generalize to unseen texts, if only to a limited extent. This confirms our hypothesis that this training model would force the encoder to learn to make embeddings based on that which unites an author’s own work and distinguishes it from others’, collectively speaking.

Nevertheless, this analysis reveals that the encoder could generalize better to unseen texts. To improve performance of this model for authorship attribution, the encoder needs to be modified so that the distribution of train text embeddings in the vector space is more similar to the distribution of test texts in the vector space. This could be achieved by producing embeddings of unseen texts that are more tightly clustered together by author or by producing embeddings of train texts that are less tightly clustered. Furthermore, given the tight clusters, it is possible the encoder may be overfitting to the training texts, enabling it to cluster these texts together more tightly at the expense of the quality of the embeddings of unseen texts, thereby explaining the general difference in the distribution of train and test embeddings.

**Conclusion**

In this research, we presented the development of a text encoder which learns to produce document vectors reflecting the author-specific (stylistic) qualities of texts. We subsequently assessed the viability of such an encoder for authorship attribution and found that the use of a bi-directional LSTM encoder with a soft voting ensemble classifier achieves a classification accuracy that surpasses all our other encoder-classifier approaches, but still fails to meet state of the art results on the same datasets.

While we failed to achieve state of the art performance,
analysis of the embeddings enabled us to confirm our hypothesis that our architecture for encoder training would produce an encoder which embeds texts based on that which unites an authors own work and distinguishes it from others, collectively speaking.

As a result, in future work, we would like to continue to improve this encoder so that it may achieve state of the art authorship attribution results. First, we would like to decrease the effect of encoder overfitting to the train set texts. This will be accomplished by modifying the training pair generation to produce significantly fewer training pairs per epoch. Furthermore, we would like to test our model on a dataset that has many training examples per author, as the CCAT-50 and CCAT-10 have only 50 examples per author, which also could have contributed to overfitting.

Second, we would like to implement and test an encoder architecture based on Transformer (Vaswani et al., 2017), to see if attention based encoder models can achieve competitive performance when dealing with textual style.

Finally, working towards our goal of creating a universal encoder, we would like to investigate the effect of training the encoder on a very large and diverse dataset, of which any potential authorship attribution dataset would be only a small subset.

Acknowledgements

We are thankful to the National Science Foundation for support through grant 1659788.

References


