

Subword Information for Authorship Attribution: A Deep Learning Approach

Anonymous ACL submission

Abstract

Authorship attribution is the process of unveiling the hidden identity of authors from a corpus of literary data. Many previous works on authorship attribution employed word-based models to capture an author’s distinctive writing style. The vocabulary of the training corpus is heavily dependent on the pre-trained word vectors, which limits the performance of these models. Alternate methods using character-based models proposed to overcome the rare word problems arising from different linguistic features fail to capture the sequential relationship of words inherently present in the texts. The question we addressed in this paper is whether it is possible to tackle the ambiguity of hidden writing style (or words) as we introduce Gaussian noise while preserving the sequential context of the text to improve authorship-related tasks. In this work, we propose two bidirectional long short-term memory (BLSTM) with a 2D convolutional neural network (CNN) over a two-dimensional pooling operation to capture sequential writing styles for distinguishing different authors. To determine the appropriate writing style representation, we used BLSTM to obtain the sequential relationship between characteristics using subword information and 2D CNN is adopted to understand the local syntactical position of the style from unlabelled input text. We extensively evaluate the model that leverages subword embedding and compare it against state-of-the-art methods for an extensive range of authors. Our methods improve 2.42%, 0.96% and 0.97% on CCAT50, Blog50 and Twitter, respectively and produce comparable results on the remaining one.

1 Introduction

Authorship attribution (AA) is the process of identifying the authors of an anonymous text according to their writing styles and characteristics, which has received increasing attention. It is made of three subtasks, which include author profiling, i.e., distinguishing the author’s demographics such as age, gender, native language, education (López-Monroy et al., 2020), authorship verification which identifies the degree of similarity of texts (Nirkhi et al., 2016) and authorship identification, given a document determine its author among a list of candidate authors (Stamatatos, 2009). In digital text forensics, AA has been widely used (e.g., attribution of proclamations to know terrorists) (Sun et al., 2012), digital humanities (e.g., attributing anonymous or disputed literary works to available authors) (Stover et al., 2016), plagiarism discovery (Foltýnek et al., 2019; Layton et al., 2015) and social media analytics (e.g., revealing multiple aliases of the same user in social media).

Over the years, authorship attribution tasks have been studied widely on a long-text document involving text samples containing thousands of words (Ramnial et al., 2016; Wanner et al., 2017) and short text such as Tweets with texts samples of few words (Schwartz et al., 2013; Azarbonyad et al., 2015; Rocha et al., 2016; Iqbal et al., 2013; Rocha et al., 2016; Seroussi et al., 2014; Koppel et al., 2011). Majority of the approaches rely on a large number of stylometric features such as length of text, the number of words, the average size of terms, the proportion of digits and capital letters, hapax-legomena, part-of-speech (POS), vocabulary richness, frequency of punctuation mark, functional words (FWS) and character/word n-gram in order to reflect both the content and the writing style of the author (Madigan et al., 2005; Aborisade and Anwar, 2018; Seroussi et al., 2014; Fabien et al., 2020). However, they can hardly capture unseen syntactic and semantic words in the texts, providing insightful meaning into the author’s writing styles.

Many traditional methods of text classification, such as bag-of-words (BOW) (Stamatatos, 2009) or simple statistics of some ordered word combinations (Cambria and White, 2014; Poria et al., 2016) have also been used. However, they fail to encode word order (Muttenzhaler et al., 2019; Luyckx and Daelemans, 2011; Sarwar et al., 2018; Alonso-Fernandez et al., 2021). Topic methods, such as LDA (Latent Dirichlet Allocation) (Blei et al., 2003) are time-consuming and inefficient as new features needs to be engineered for large scale datasets for
AA tasks (Agrawal et al., 2018; Seroussi et al., 2014; Mikolov et al., 2013b; Modupe et al., 2014).

As opposed to topic modelling, deep learning neural network models, e.g., convolution neural network (CNN) (Kalchbrenner et al., 2014; Kim et al., 2016), RNN with 1D max-pooling (Lai et al., 2015) or attention-based operation (Wu et al., 2021) employs pretrained word embedding features as inputs to achieve better performance by mapping vector space to extract semantics features over the time-step of sentences (Ding et al., 2017; Sari et al., 2017; Gómez-Adorno et al., 2018; Bagnall, 2015; Gupta et al., 2017; Jafariakinabad et al., 2019). However, these models generally converge slowly. In addition, if embedding vectors of rare words are poorly estimated, it would likely harm the representations of words surrounding the author’s writing style and the model’s performance (or the classification models). This is particularly problematic in morphologically rich languages with long-tailed frequency distributions or domains with dynamic vocabularies (e.g., short digital text).

In this work, we introduce bidirectional long short-term memory (BLSTM) with 2D convolution and 2D max-pooling operation that employs byte-pair encoding (BPE) to transform input texts into subword embedding. First, we use BLSTM layers to capture inherent semantic features on both the time-step dimension and the subword feature dimension. And then feed the feature vectors into a 2D CNN and 2D max-pooling to obtain more local syntactical information to represent the input text for AA tasks.

The reminder of the paper is organized as follows. In Section 2, we provide an overview of related work. In Section 3, we describe the structure of our proposed BLSTM-CNN max-pooling model for AA tasks in details. In Section 4, we described the details about the dataset, hyperparameters setting and the experimental results. Finally, Section 5 depicts our conclusions as well as future work.

2 Related Work

Traditionally, the AA task largely relies on extracting stylometric features related to content or style to ascertain the writer of the text. Most of the existing methods based on stylometric features aim to capture writing patterns at different linguistic categories, e.g., lexical, syntactic, structure and semantics. Lexical features capture an individual’s characters and words to describe vocabulary richness and choice for particular symbols or words. At the character level, features include the number or frequency of different characters. The word-level feature combines the total number of words, average word length, a portion of short/long words, most common words, and the number of unique words (Argamon and Levitan, 2005; Juola, 2007; Stamatatos, 2009).

Another practical set of characteristics combined with lexical is the n-grams. They describe sequences of n elements next to one another. The elements can be different, for example, a sequence of characters, words, symbols, and syllables. However, as the dimensionality of the n-gram vector space grows with n, the character or word n-gram features capture too much content-specific rather than related stylistic information (Alonso-Fernandez et al., 2021). For example, Plakias and Stamatatos (2008) used tensors of the second order with 2500 most frequent 3-grams to represent stylistic components for a given texts and Muttenzhaler et al. (2019) show the influence of punctuation marks with n-gram model for AA tasks, while masking punctuation marks with asterisk (*) symbol. Markov et al. (2017) allege that digits and named entities are other critical features for selecting writing styles. Sapkota et al. (2015) connect the potential of using character-level n-gram features to the high priority of subword features (e.g. suffixes and prefixes) in authorship-related tasks. Zhao et al. (2019) relax the constraint of n-gram features and analyze the co-occurrence of word pairs instead. As text representations created from n-gram model tend to be high-dimensional and sparse, Niu et al. (2017) employ principal component analysis (PCA) to decrease them into low-dimensional vectors and both Seroussi et al. (2011) and Zhou et al. (2018) is the nearest collaborative approach that utilises novel models of finding the style similarity based on topic models. Structural features capture the organization of paragraphs and sentences. They include the number of sentences, paragraphs, lines, punctuation, average length of sentences and paragraphs. With structure features, elements such as greetings and signatures in a text can be analyzed. Koppel et al. (2011) used lexical and structural features with multiple randomized characteristics to unveil the writing similarity between two authors by ignoring the order of word, syntax, or meaning in the text. Recently, many researchers have turned to neural language models such as the skip-gram model (Mikolov et al., 2013a,b; Pennington et al., 2014) based on the distributed representation of the words to learn the distribution of the writing style (Ding et al., 2017; Gómez-Adorno et al., 2018; Posadas-Durán et al., 2017).

On the other hand, syntactic features characterize the use of punctuation and function words (FWS), which help define the relationship of elements in a sentence. It also includes POS tagging by categorizing a word as either verb, noun, pronoun, adjective (e.g., according to its function). Bevendorff et al. (2019) develop character trigram vectors for the documents and evaluate the variations between each couple of documents as features using seven distance measures. Bagnall (2015) employs a recurrent neural network (RNN) model on character level to verify authorship and obtain a higher accuracy than Bevendorff et al. (2019), proving the power of deep neural networks on authorship-based tasks. Sari et al. (2017) worked on using continuous representations via a neural network jointly with the classification layer for authorship attribution, and Shrestha et al. (2017) performed authorship attribution of short digit text (e.g.,
tweets) using CNNs over character n-grams by estimating the importance of input text fragments to improve model interpretability. Zhang et al. (2018) applied a novel strategy to encode the syntax parse tree of the sentence into a learnable distributed representation. Specifically, they build an embedding vector for each word in the text by encoding the path as a syntax tree corresponding to the word. An attribution by Sari et al. (2018) is one of the state-of-the-art feature-based techniques that extract features using various stylometric features and achieves excellent performance. However, it does not take full advantage of the semantic features. Yao et al. (2019) is the convolutional graph network for text classification but can not run on IMDb62 datasets because a huge text graph takes up too much computer memory. Jafarianinbad et al. (2019) evaluation the strength and robustness of the syntactic recurrent neural network to encode the syntactic patterns of a document in a hierarchical structure for AA tasks. However, it is not clear what is captured by the learned vectors. We leverage and handle essential elements employing the BLSTM-CNN module feed with subword embedding based on the BPE algorithm (Sennrich et al., 2016) to help capture syntactic and sequential semantic information from the unlabelled text for AA tasks.

3 Proposed method

The first portion of our system, as presented in Fig. 1 adopt a byte-pair-encoding (BPE) algorithm as an embedding tactic to transform the pure text into numerical representations. In the second phase, we feed the embedding modules into a bidirectional LSTM to understand the underlying semantic and apply CNN max-pooling overtime to capture the local spatial syntactical position on writing style from the input text. The classification consists of a fully-connected layer and soft-max function, which is sufficient to fit the function which takes the features and outputs the classification result. Besides, we combined annealed Gaussian noise with training the model to learn the writing style representations for AA tasks, which helped avoid overfitting and achieved lower training loss. We evaluated the model’s performance using a k-fold cross-validation with the Twitter, blog, review, novel, and essay datasets.

3.1 Sub-word Embedding

BPE is a tokenization technique adopted in machine translation to deal with imaginary word problems or hidden writing in a given text. It is unsupervised and requires no information about the author. The algorithm of BPE first initializes a symbol by splitting the input text into characters. Then, iteratively count all symbol pairs and replace each occurrence of the most frequent pair \((x,y)\) with a new \(xy\) symbol and add it to the symbol set named “merge operation”. Each merge operation generates a new symbol. The size of the final symbol set is equal to that of the first single character, plus the number of merge-operations. The only hyperparameter for the BPE algorithm, as shown in Algorithm 1 is the number of the merge operation and return a meaningful trait (i.e., the word if the merge operation is large).

Algorithm 1: BPE algorithm

Input: training data \(D\) of words split into character sequence with number \(N\) of rules

Output: list of \(K\) of \(N\) merge rules

\[K := []\]

\[\text{while} \ \text{length}(K) \leq N \ \text{do}
\]

\[
(x, y) := \text{argmax}\{\text{count}(x, y)\};
\]

\[
\text{rule} := (x, y) \rightarrow xy;
\]

\[D := \text{apply} \ (\text{rule}, D);
\]

\[K := \text{append} \ (\text{rule}, K);
\]

\[\text{return} \ K\]

For untokenized unique text, we first split it into a single character and then iteratively do the merge operations following the merge order in the training step; until there are no more symbols that can be merged. That is to say, if the number of merge operations is large, the token will tend to have more characters, and the granularity tends to be large. Otherwise, the granularity of the original text will be small. In our system, we do not use BPE as a compression algorithm. Instead, we use this algorithm to find sub-words as n-grams with high frequencies for word segmentation, achieved if we joined characters together. However, we did not substitute them with new symbols. An example of how subwords are obtained from a raw input text after \(N\) iterations is shown in Table 1.

The text has now been subdivided into subword sequences. To use subword embedding to represent the text, we first create a one-hot vector for each subword type. The one-hot vector for the \(i\)th subword in the vocabulary is a sparse binary vector \(o_i\) which has 1 as the \(i\)th element and all 0 for others. After that, we project this embedding hyperspace onto a smaller hyperspace by multiplying the one-hot embedding with a subword embedding matrix \(S\) with size \(N \times D\), where \(N\) represent the sub-word vocabulary size and \(D\) is the dimension for the target embedding hyperspace. Therefore, we represent each sub-word information as a dense vector \(s_i = S^T o_i\), and the text with length \(T\) is represented by a sequence of subword embedding vectors \(\{s_1, s_2, \ldots, s_T\}\). Therefore, the subword embedding matrix \(S\) are trained together.

3.2 Feature Extraction

The second part of our system is feature extraction based on the sub-word embeddings feeds into BLSTM and CNN module, and the final output was a neuron representing the probability of feature vectors belonging to certain authorship. Given the subword embedding, we use BLSTM to find the inherent grammatical relationship in the author’s writing style embedded in the
The idea is to introduce an adaptive gating mechanism, which decides the degree to keep the previous state and the output gate. Finally, $b_f$, $b_i$, and $b_o$ are the bias for the forget, input and output gate. $\sigma$ denotes the logistic sigmoid function. The memory cell can be calculated as:

$$\hat{c}_t = \tanh(x_t W_{x,c} + h_{t-1} W_{h,c} + b_c)$$  \(2\)

where $W_{x,c}$ and $W_{h,c}$ represent the weights of the LSTM from input to the memory and from the hidden state to the memory, respectively, and the $b_c$ denotes the bias. The memory cell at the time step $t$ was computed by

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$  \(3\)

where $\odot$ denoted element-wise multiplication. The hidden state can be updated as:

$$h_t = i_t \odot \tanh(c_t)$$  \(4\)

The RNN model was forward and the output at the time steps $t$ depends on the past context as well as the hidden state, e.g. the future context. Schuster and Paliwal (1997) introduced a BLSTM to extend the unidirectional LSTM by introducing a second hidden layer, where the hidden connections flow in opposition temporal order. Therefore, the model is able to exploit information from both the past and the future. In this study, BLSTM is used to capture the past and the future writing style information. As shown in Fig. 1, our system contains two sub-networks for the forward (f) and backward (b) sequences context based on the subword embedding from the input text at each time step $t$ as follow:

$$h^f_t = \sigma(x_t W^f_{x,h} + h^f_{t-1} W^f_{h,h} + b^f_h)$$  \(5\)

$$h^b_t = \sigma(x_t W^b_{x,h} + h^b_{t-1} W^b_{h,h} + b^b_h)$$

The output at each time $t$ can be computed as:

$$h_t = [h^f_t \odot h^b_t] W_{h,o} + b_o$$  \(6\)

where $\odot$ is the element-wise sum to combine the forward and backward pass outputs.
3.2.2 Convolutional Layer

From the BLSTM layer, we have access to the future context as well as the past context, $\theta_t$ is related to all other writing style (or words) in the given text. In this study, we effectively treat the matrix as a feature vectors, so 1D convolution and the max pooling operation were used to capture local syntactical information.

For matrix $H_t = [h_{11}, h_{21}, \ldots, h_{l1}]$, $H \in \mathbb{R}^{1 \times d_w}$ obtained from BLSTM layer, where $d_w$ is the size of the subword embeddings. Then narrow convolution is utilized to extract local features information over $H$. The convolution operation involves a filter $m \in \mathbb{R}^{k \times d}$, which is applied to a window of $k$ subwords and d feature vectors. For instance, a feature $o_{i,j}$ is generated from a window of vectors $H_{i+k-1,j+d-1}$ as:

$$o_{i,j} = f(m \cdot H_{i+k-1,j+d-1} + b)$$

where $i$ ranges from 1 to $(l-k+1)$, $j$ ranges from 1 to $(d_w - d + 1)$, $(\cdot)$ represents dot product, $b \in \mathbb{R}$ is a bias and $f$ is a non-linear function similar to hyperbolic tangent. So, we applied the filter to each possible window of the BLSTM layer matrix $H$ to obtain a feature map $O$:

$$O = [o_{1,1}, o_{1,2}, \ldots, o_{1-k+1,d_w-d+1}]$$

with $O \in \mathbb{R}^{(l-k+1) \times (d_w-d+1)}$ represent one convolution filter process. The convolution layer have multiple filters for the same size filter to learn complementary features, or multiple kinds of filter with different size. Then 2D max pooling operation $p \in \mathbb{R}^{p_1 \times p_2}$ utilizes to find a fixed length vector by applying it to each possible window of matrix $O$ to extract the maximum value:

$$p_{i,j} = \text{down}(O_{i+p_1-1, j+p_2-1})$$

where $\text{down}()$ represents the max pooling operation function, $i = [1, 1+p_1, \ldots, 1+(l-k+1/p_1-1) \cdot p_1]$ and $j = [1, 1+p_2, \ldots, 1+(d_w - d + 1/p_2 - 1) \cdot p_2]$.

Then, we compute the pooling operation as follow:

$$h^* = [p_{1,1}, p_{1,1+p_1-1}, \ldots, p_{1,1+(l-k+1/p_1-1) \cdot p_1}, 1 \cdot (d_w - d + 1/p_2 - 1) \cdot p_2]$$

where $h^* \in \mathbb{R}$, and the length of $h^*$ is $[1 \cdot (l-k+1/p_1) \times [d_w - d + 1/p_2 - 1]]$.

3.2.3 Classification Layer

For the AA tasks, the output $h^*$ from the max pooling was passed over the fully-connected layer of the input text $X$, then feeds it to a softmax function as a classifier to predict the inherent writing style related to a particular candidate author $\hat{y}$ from the set of discrete set of author (or classes) $Y$. So, the classifier takes the hidden state $h^*$ as input as follows:

$$\hat{y} = \text{argmax}_y \hat{p}(y|x)$$

To learn the model parameters we minimize the cross-entropy loss as the training objective by calculating the loss as a regularized sum:

$$L = -\frac{1}{m} \sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2$$

where $t \in \mathbb{R}^m$ represent the one-hot encoding for the ground truth values, $y \in \mathbb{R}^m$ is the predicted probability of the candidate author by softmax, $m$ is the number of expected target authors, and $\lambda$ is the $\ell_2$ regularization parameter. Training is done through the Adam Optimization algorithm (KingaD, 2015) as further explained in Section 4.2. Finally, the pseudocode of our model is given in Algorithm 2, where we use simplified variables to make the procedure clear.

4 Experiment and analysis

4.1 Datasets

We benchmark our model by experimenting on three openly available datasets covering a large spectrum of authorship attribution on CCAT50, IMDb62, Blog50 and new Twitter datasets. The first three datasets have been used for many previous studies (Sari et al., 2017, 2018; Seroussi et al., 2014; Zhang et al., 2018). At the same time, the Twitter dataset has also been used by (Ruder et al., 2016; Shrestha et al., 2017) extensively. Due to the limitation of Twitter policy, the actual content of tweets we were omitted; however, the available users’ IDs and the tweet IDs enable us to collect relevant tweets. Table 2 shows some detailed statistical information. CCAT50 has a total of 5,000 documents written by 50 authors (Stamatatos and Koppel, 2011). IMDb62 comes from Internet Movie Database (IMDB) containing 62,000 movie reviews and 17,550 message posts from 62 prolific authors. In this paper, we choose 62,000 movie reviews as the dataset doing experiments (Seroussi et al., 2014). Blog50 original contains 681,288 posts by 19,320 bloggers, and in this paper, we select posts written by the top 50 bloggers. Twitter was an influencer dataset from a list of 4,391 celebrities, such as columnists, musicians and influencers on social media in 68 areas covering politics, social unrest and tech to arts and culture for AA tasks. We collected over a million tweets for these users in August and September 2019 using python Twitter API (Gupta et al., 2017). For our experiment, each dataset is split by sampling 60% of each author’s documents into a training set, 20% for validation and renders remainder for testing over 10-fold cross-validation as used in most AA tasks.

4.2 Experiment settings

We used Adam optimization (KingaD, 2015) for small-batch training. The default mini-batch size is 64 due to constraints on the graphics processor (GPU) as NVIDIA
**Algorithm 2:** Pseudocode for BLSTM-2DCNN max-pooling with subword information

**Input:** Training data \( X = \{X_i\}_{i=1}^m \) represent the batch size, \( n \) is the number of training samples, \( w \) is model parameters and \( l \) represent the length of the input text.

**Output:** trained model.

1. Randomly initialize \( w \);
2. **foreach each iteration do**
   3. **forall** \( k \in \{1, 2, \cdots, \left\lfloor \frac{n}{w} \right\rfloor \} \) do
      4. Sample from each batch \( X_{lk} \) from \( X \);
      5. Divide each sample in the batch into the sequence \( \{X_{1,k}, X_{2,k}, \cdots, X_{l,k}\} \);
      6. Feed sequential batch into BLSTM consisting of forward and backward neuron, respectively, and
         output two output sequence \( \{h_{1,k}, h_{2,k}, \cdots, h_{1,k}, h_{2,k}, h_{3,k}, \cdots, h_{5,k}\} \) Equation 5;
      7. Concatenate the BLSTM layer to obtained \( H \in \mathbb{R}^{l \times d} \), the narrow convolution is used to extract local
         dependent features over \( H \) to produce a feature map \( O \);
      8. Then, \( p \in \mathbb{R}^{P_1 \times P_2} \) is applied to each possible window of matrix \( O \) to obtain \( h^* \) Equation 10
         represent the stylometric representation (e.g., writing styles) of the input \( X \);
      9. Feed the output \( h^* \) into the softmax classifier layer and obtain the classification result in
         Equation 11 and 12 respectively;
   10. Update \( w \) by minimizing with the cross-entropy loss in Equation 13 using Adman algorithm (KingaD,
       2015);

---

**Table 2:** Statistics of the datasets. \( c \): Number of authors. \( D \): Number of documents per authors. \( \mu \): average number of words per \( D \). \( \sigma \): average number of character per \( D \). \( \omega \): maximum number of character. \( \kappa \): minimum number of characters.

<table>
<thead>
<tr>
<th>Data</th>
<th>( c )</th>
<th>( D )</th>
<th>( \mu )</th>
<th>( \sigma )</th>
<th>( \omega )</th>
<th>( \kappa )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAT50</td>
<td>50</td>
<td>1000</td>
<td>584</td>
<td>3010</td>
<td>8716</td>
<td>345</td>
</tr>
<tr>
<td>IMDB62</td>
<td>62</td>
<td>62000</td>
<td>345</td>
<td>1742</td>
<td>11617</td>
<td>82</td>
</tr>
<tr>
<td>Blog50</td>
<td>50</td>
<td>19320</td>
<td>440</td>
<td>541</td>
<td>30712</td>
<td>3</td>
</tr>
<tr>
<td>Twitter</td>
<td>1350</td>
<td>4391</td>
<td>31</td>
<td>229</td>
<td>30750</td>
<td>19</td>
</tr>
</tbody>
</table>

GeForce 2080Ti to train the model with 0.001 as the initial learning rate and utilize ReduceLRonPlateau schedule with the patience of 5 epochs and a decay factor of 0.5. During training, the dimension of the subword vector is 300. The hidden units of BLSTM are 128. We use 100 convolutional filters for the window sizes of (3, 3) with a max-pooling size of (2, 2). For regularization, we employ Dropout operation (Hinton et al., 2012) with a dropout rate of 0.5 for the subword embeddings, 0.3 for the BLSTM and 0.2 for the penultimate layer with Gaussian Noise of 0.2 active at the training time. We also use \( \ell_2 \) penalty with coefficient \( 10^{-3} \) over the parameter and trained for 20 epochs. All word vectors and feature vectors are randomly initialized and learned, and updated during the training process. The dimensions of the word vector and hidden layer size are \( d = 64 \) in all models. We use 128 convolutional filters, each for window sizes of (3, 3) and 2D pooling sizes of (2, 2). All experiments in this paper were repeated five times with three random seeds (2020), and the accuracy in this paper refers to the average classification accuracy.

### 4.3 Results and Discussion

In this work, we implements three different model, BLSTM-2DCNN, BLSTM-2DCNN word embedding and BLSTM-2DCNN gradient noise with subword information. Table 3 presents the performance of all the three models and other state-of-the-art models on four dataset for authorship-based tasks. The overall authorship attribution accuracies of our methods and the baseline are provided in Table 3. The "(−)" indicate that the feature and the model are excluded. As shown in Table 3, the BLSTM-2DCNN+Gaussian noise with subword embedding achieves comparative performance on three out of four datasets. Gaussian noise was combined with \( \ell_2 \) regularization to gain roughly 10% better performance when compared to both traditional methods and the existing CNN based models. Essentially, it achieves 2.9%, 1.6% and 0.9% test accuracy on CCAT50, Blog50 and Twitter datasets, respectively. In addition, the performance of the proposed model is superior to that of the CNN and BertAA model (Ruder et al., 2016; Fabien et al., 2020), which shows that learning from characters or leveraging on the pre-trained language model without feature engineering task can help to improve the performance for AA tasks. Our method is much better than the BertAA model, which validates the effectiveness of integrating a pre-trained BERT (Devlin et al., 2018) language model with an extra dense layer to perform authorship classification. In addition, different from existing CNN-based methods, we leveraged the extracted features employing the BPE algorithm to represent words (e.g., the writing style) by its index in the vocabulary together with its subword vector classes. Consequently, the proposed model inherits the advantage of both traditional CNN-LSTM model (Ruder et al., 2016; Gupta et al., 2017; Jafariakinabad et al., 2019) and
models and the Gaussian noise, where hashtags or emoticons are the most characteristic features. When authors review similar movies, the BertAA model with 50 authors. Differences for the IMDb62 domain are boosted on IMDb62 because they are less sensitive to topical divergence. They are, however, less helpful in short digit text, e.g., Blog50 and Twitter domain, where hashtags or emoticons are the most characteristic features.

To further substantiate the effectiveness of our model, we tested CNN and BertAA models and the Gaussian noise, respectively. We then reported the performance of the results in Table 4. For the CNN-based and BertAA model, we add Gaussian noise before the softmax classifier on the same network structure. Comparing CNN-based and BertAA models, we see that each model can improve authorship classification accuracy using the same extracted features from BPE algorithms. In addition, it can be seen in Table 4 that our model is superior to the counterpart CNN-based or BertAA model with a pre-trained weighted vector. The accuracy and convergence curves (e.g., loss) on the datasets were displayed in Fig. 2, respectively. In Fig. 3, we can see the best classification accuracy with faster convergence speed in the training process for Twitter datasets compared to the CNN-char model (Ruder et al., 2016).

5 Conclusion

This paper demonstrates that input embedded vectors employing subword information feeds with the BLSTM-2DCNN model could learn stylometric representations of different linguistic modalities for AA tasks. It showed values such a configuration’s effectiveness in dealing with common spelling errors from unstructured texts due to orthography and phonetic reasons, then learns stylistic and topical information to classify the author. In addition, the Gaussian noise is introduced to the fully conventional layers, which substantially reduces the large number of parameters arising from the model structure. Thus, the convergence rate of the model significantly speeds up and improve the classification accuracy. We evaluated the model against the state-of-the-art methods for an extensive range of authors, demonstrating the proposed model’s effectiveness in handling morphological variance and is applicable across authorship-related tasks. Future works will explore combining the model with a self-attention mechanism to model different linguistic levels (e.g., structure, POS tagging, dependency and semantics) applying subword information to improve the alignment of words in the input texts during training in style-related tasks to find ways to advance the research on authorship-based tasks.

Table 3: Performance comparison and accuracy scores on four mainstream AA datasets.

<table>
<thead>
<tr>
<th>Models</th>
<th>CCAT50</th>
<th>IMDb62</th>
<th>Blog50</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM with 3-gram (Plakias and Stamatatos, 2008)</td>
<td>67.00</td>
<td>81.40</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Imposters (Koppel et al., 2011)</td>
<td>–</td>
<td>76.90</td>
<td>26.00</td>
<td>52.50</td>
</tr>
<tr>
<td>LDAH-S with topics (Seroussi et al., 2011)</td>
<td>–</td>
<td>72.00</td>
<td>18.30</td>
<td>38.30</td>
</tr>
<tr>
<td>SVM Affix + punctuation (Sapkota et al., 2015)</td>
<td>69.30</td>
<td>69.90</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Style, content &amp; hybrid (Sari et al., 2018)</td>
<td>–</td>
<td>75.76</td>
<td>84.51</td>
<td>–</td>
</tr>
<tr>
<td>CNN-character (Ruder et al., 2016)</td>
<td>–</td>
<td>91.70</td>
<td>49.40</td>
<td>86.80</td>
</tr>
<tr>
<td>CNN-word (Ruder et al., 2016)</td>
<td>–</td>
<td>84.30</td>
<td>43.00</td>
<td>80.50</td>
</tr>
<tr>
<td>CNN n-gram (Shrestha et al., 2017)</td>
<td>76.50</td>
<td>95.21</td>
<td>53.09</td>
<td>–</td>
</tr>
<tr>
<td>Continuous n-gram (Sari et al., 2017)</td>
<td>72.60</td>
<td>95.12</td>
<td>52.82</td>
<td>–</td>
</tr>
<tr>
<td>Syntax-CNN (Zhang et al., 2018)</td>
<td>81.00</td>
<td>96.16</td>
<td>56.73</td>
<td>–</td>
</tr>
<tr>
<td>LSTM word embedding (Gupta et al., 2017)</td>
<td>61.47</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>GRU word embedding (Gupta et al., 2017)</td>
<td>69.20</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Syntactic RNN word &amp; POS (Jafariakinabad et al., 2019)</td>
<td>76.72</td>
<td>92.15</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BertAA (Fabien et al., 2020)</td>
<td>–</td>
<td>90.70</td>
<td>59.70</td>
<td>–</td>
</tr>
<tr>
<td>BLSTM-2DCNN</td>
<td>73.25</td>
<td>81.25</td>
<td>48.15</td>
<td>49.52</td>
</tr>
<tr>
<td>BLSTM-2DCNN word embedding</td>
<td>74.50</td>
<td>89.72</td>
<td>53.26</td>
<td>79.30</td>
</tr>
<tr>
<td>BLSTM-2DCNN subword+Gaussian noise</td>
<td>83.42</td>
<td>93.72</td>
<td>60.67</td>
<td>87.76</td>
</tr>
</tbody>
</table>

References


<table>
<thead>
<tr>
<th>Models</th>
<th>CCAT50</th>
<th>IMDb62</th>
<th>Blog50</th>
<th>Twitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-char (Ruder et al., 2016)</td>
<td>76.77</td>
<td>91.01</td>
<td>59.70</td>
<td>84.29</td>
</tr>
<tr>
<td>CNN-word (Ruder et al., 2016)</td>
<td>77.03</td>
<td>92.30</td>
<td>62.40</td>
<td>82.50</td>
</tr>
<tr>
<td>BertAA (Fabien et al., 2020)</td>
<td>78.90</td>
<td>93.00</td>
<td>64.40</td>
<td>62.50</td>
</tr>
<tr>
<td>Our method</td>
<td>83.42</td>
<td>93.72</td>
<td>60.67</td>
<td>87.76</td>
</tr>
</tbody>
</table>

Table 4: Performance comparison of CNN and BertAA model on all the datasets.

Figure 2: The accuracy and convergence curve over iterations on all the datasets

Figure 3: The accuracy and convergence curve over iterations on Twitter dataset.
Transactions on Biometrics, Behavior, and Identity Science.


Lukas Muttenzhalter, Gordon Lucas, and Janek Amann. 2019. Authorship attribution in fan-fictional texts given variable length character and word n-grams. In *CLEF (Working Notes).*


