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 (Foltỳnek 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 fails to encode word order (Muttenthaler 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 pre-trained 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 provides 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 spaces 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 Muttenthaler 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 untilizes 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. Jafariakinabad 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 := [ ];
while length (K) ≤ N do
    (x, y) := argmax_(x,y) {count_D(x, y)};
    rule := ⟨(x, y) ⟶ xy⟩;
    D := apply(rule, D);
    K := append(rule, K);
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 $\mathcal{N}$ 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 $\mathcal{S}$ with size $\mathcal{N} \times \mathcal{D}$, where $\mathcal{N}$ represent the sub-word vocabulary size and $\mathcal{D}$ is the dimension for the target embedding hyperspace. Therefore, we represent each sub-word information as a dense vector $s_i = \mathcal{S}_o^T i$, and the text with length $\mathcal{T}$ is represented by a sequence of subword embedding vectors $(s_1, s_2, \cdots, s_T)$. Therefore, the subword embedding matrix $\mathcal{S}$ 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

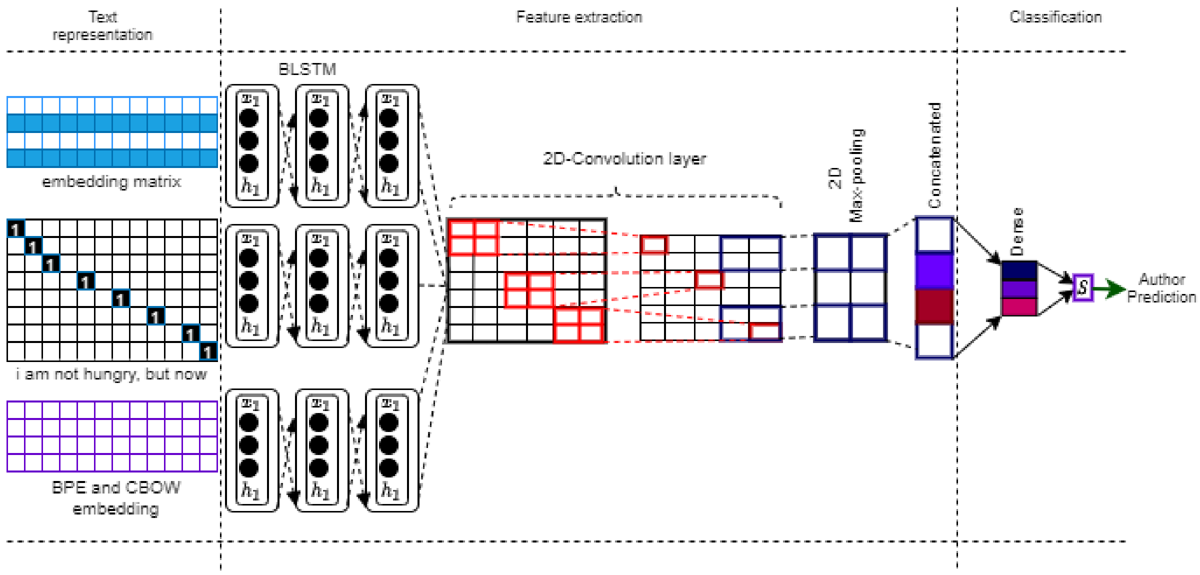


Figure 1: BLSTM-CNN model with subword information

Iteration	Words
0	workers work in workshop
1	workersworkinworkshop
2	workersworkinworkinshop
3	wor kerswor kinwor kshop
4	work ers work in work shop
N	work er s work in work shop

Table 1: BPE example given raw input text.

subword and the CNN to capture the local syntactical information from the input text.

3.2.1 Bidirectional LSTM Layer

Long short-term memory (LSTM) was first proposed by (Schuster and Paliwal, 1997) to overcome the gradient vanishing problem of recurrent neural network (RNN). The idea is to introduce an adaptive gating mechanism, which decides the degree to keep the previous state and memorize the extracted features of the current input text. So, given a sequence of input text $X = \{x_1, x_2, \cdots, x_l\}$, where l represent the length of the input text, LSTM processes the input as a subword information. At each time-step t, the memory c_t and the hidden state h_t are updated as following:

$$\begin{aligned} f_t &= \sigma(x_t W_{x,f} + h_{t-1} W_{h,f} + b_f) \\ i_t &= \sigma(x_t W_{x,i} + h_{t-1} W_{h,i} + b_i) \\ o_t &= \sigma(x_t W_{x,o} + h_{t-1} W_{h,o} + b_o) \end{aligned} \quad (1)$$

where $W_{x,f}$ and $W_{h,f}$ are the weights of the LSTM from input to forget gate and from the hidden state to the forget gate. The $W_{x,i}$ and $W_{h,i}$ are the weights from input to input gate and from the hidden state to the input gate. Similarly, $W_{x,o}$ and $W_{h,o}$ represent the weights from input to output gate and from the hidden state to the output gate. Finally, b_f, b_i, and, b_o are the bias for the forget, input and output gate. σ denotes the logistic sigmoid function. The memory cell can be calculated as:

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

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

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

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

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

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) sequence context based on the subword embedding from the input text at each time step t as follow:

$$\begin{aligned} h_t^f &= \sigma(x_t W_{x,h}^f + h_{t-1}^f W_{h,h}^f + b_h^f) \\ h_t^b &= \sigma(x_t W_{x,h}^b + h_{t-1}^b W_{h,h}^b + b_h^b) \end{aligned} \quad (5)$$

The output at each time t can be computed as:

$$h_t = [h_t^f \otimes h_t^b] W_{h,o} + b_o \quad (6)$$

where $\otimes$ 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, o_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 $H_t = \{h_1, h_2, \cdots, h_l\}$, $H \in \mathbb{R}^{l \times d^w}$ 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 $H_{i:i+k\text{-}1, j:j+d\text{-}1}$ as

$$o_{i,j} = f(\boldsymbol{m} \cdot H_{i:i+k\text{-}1,\, j:j+d\text{-}1} + b) \qquad (7)$$

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 an 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**:

$$O = [o_{1,1}, o_{1,2}, \cdots, o_{l-k+1, d^w-d+1}] \qquad (8)$$

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 $p \in \mathbb{R}^{(p_1 \times p_2)}$ is utilizes to obtain a fixed length vector by applying it to each possible window of matrix O to extract the maximum value:

$$p_{i,j} = down(O_{i:i+p_1, j:j+p_2}) \qquad (9)$$

where $down(\cdot)$ represents the map pooling operation function, $i = (1, 1+p_1, \cdots, 1+(l-k+1/p_1-1)\cdot p_1)$ and $j = (1, 1+p_2, \cdots, 1+(d_w-d+1/p_2-1)\cdot p_2)$. Then, we compute the pooling operation as follow:

$$h^* = [p_{1,1}, p_{1,1+p_2}, \cdots, p_1 + (l-k+1/p_1-1)\cdot p_1, \\ 1 + (d_w - d + 1/p_2 - 1)\cdot p_2] \qquad (10)$$

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

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:

$$\hat{p}(y|x) = softmax\big(W^{(x)}h^* + b^{(x)}\big) \qquad (11)$$

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, \hat{p}(y|x) \qquad (12)$$

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

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2 \qquad (13)$$

where $\boldsymbol{t} \in \mathbb{R}^m$ represent the one-hot encoding for the ground truth values, $\boldsymbol{y} \in \mathbb{R}^m$ is the predicted probability of the candidate author by softmax, m is the number of expected target authors, and λ is the ℓ_2 regualrization 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 $\mathbf{X} = \{\mathrm{X_i}\}_{\mathrm{i}=1}^{\mathrm{n}}$ 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.

Randomly initialize w;
foreach *each iteration* **do**
  forall $k \in \{1, 2, \cdots, \lfloor \frac{\mathrm{n}}{\mathrm{m}} \rfloor\}$ **do**
    Sample from each batch $\mathbf{X}_k$ from $\mathbf{X}$;
    Divide each sample in the batch into the sequence $\{\mathbf{X}_k^1, \mathbf{X}_k^2, \cdots, \mathbf{X}_k^{\mathrm{l}}\}$;
    Feed sequential batch into BLSTM consisting of forward and backward neuron, respectively, and outcome two output sequence $\{\boldsymbol{h}_{\mathrm{fk}}^1, \boldsymbol{h}_{\mathrm{fk}}^2, \cdots, \boldsymbol{h}_{\mathrm{fk}}^{\mathrm{l}}, \boldsymbol{h}_{\mathrm{bk}}^1, \boldsymbol{h}_{\mathrm{bk}}^2, \cdots, \boldsymbol{h}_{\mathrm{bk}}^{\mathrm{l}}\}$ Equation 5;
    Concatenate the BLSTM layer to obtained $H \in \mathbb{R}^{\mathrm{l} \times \mathrm{d}^w}$, the narrow convolution is used to extract local dependent features over H to produce a feature map O ;
    Then, $\mathrm{p} \in \mathbb{R}^{(\mathrm{p}_1 \times \mathrm{p}_2)}$ 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 $\mathbf{X}$;
    Feed the output h^* into the softmax classifier layer and obtain the classification result in Equation 11 and 12 respectively;
    Update w by minimizing with the cross-entropy loss in Equation 13 using Adman algorithm (KingaD, 2015);

Data	c	D	μ	σ	ω	κ
CCAT50	50	1000	584	3010	8716	345
IMDb62	62	62000	345	1742	11617	82
Blog50	50	19320	440	541	30712	3
Twitter	1350	4391	31	229	30750	19

Table 2: Statistics of the datasets. c: Number of authors. D: Number of documents per authors. μ: average number of words per D. σ: average number of character per D. ω: maximum number of character. κ: minimum number of characters.

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 ℓ_2 penalty with coefficient 10^{-5} 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 $\mathrm{d} = 64$ 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 ℓ_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	CCAT50	IMDb62	Blog50	Twitter
SVM with 3-gram (Plakias and Stamatatos, 2008)	67.00	81.40	–	–
Imposters (Koppel et al., 2011)	–	76.90	26.00	52.50
LDAH-S with topics (Seroussi et al., 2011)	–	72.00	18.30	38.30
SVM Affix + punctuation (Sapkota et al., 2015)	69.30	69.90	–	–
Style, content & hybrid (Sari et al., 2018)	–	75.76	84.51	–
CNN-character (Ruder et al., 2016)	–	91.70	49.40	86.80
CNN-word (Ruder et al., 2016)	–	84.30	43.00	80.50
CNN n-gram (Shrestha et al., 2017)	76.50	95.21	53.09	–
Continuous n-gram (Sari et al., 2017)	72.60	95.12	52.82	–
Syntax-CNN (Zhang et al., 2018)	81.00	**96.16**	56.73	–
LSTM word embedding (Gupta et al., 2017)	61.47	–	–	–
GRU word embedding (Gupta et al., 2017)	69.20	–	–	–
Syntactic RNN word & POS (Jafariakinabad et al., 2019)	76.72	92.15	–	–
BertAA (Fabien et al., 2020)	–	90.70	59.70	–
BLSTM-2DCNN	73.25	81.25	48.15	49.52
BLSTM-2DCNN word embedding	74.50	89.72	53.26	79.30
BLSTM-2DCNN subword+Gaussian noise	**83.42**	93.72	**60.67**	**87.76**

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

BPE algorithms with Gaussian noise, which contributes to the performance improvement for the AA dataset as shown in Table 3.

Our model outperforms all the CNN variants by more than 10% and 4% for both Blog and Twitter datasets with 50 authors. Differences for the IMDb62 domain render less discriminatory words or character sequences when authors review similar movies. The BertAA model is 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 showcases 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.

References

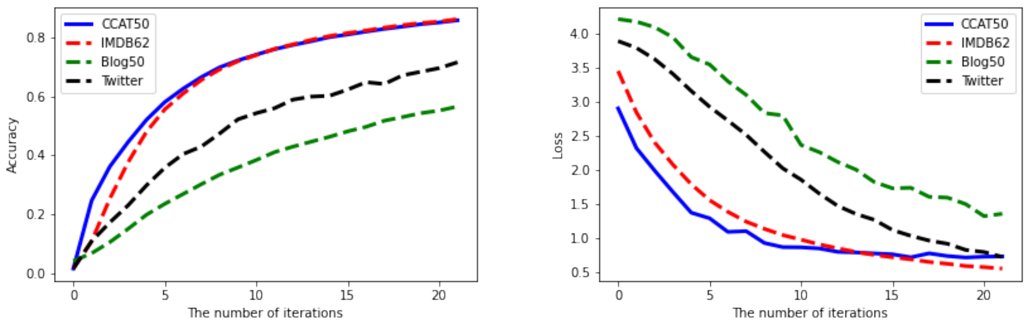
Opeyemi Aborisade and Mohd Anwar. 2018. Classification for authorship of tweets by comparing logistic regression and naive bayes classifiers. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 269–276. IEEE.

Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88.

Fernando Alonso-Fernandez, Nicole Mariah Sharon Belvisi, Kevin Hernandez-Diaz, Naveed Muhammad, and Josef Bigun. 2021. Writer identification using microblogging texts for social media forensics. *IEEE*

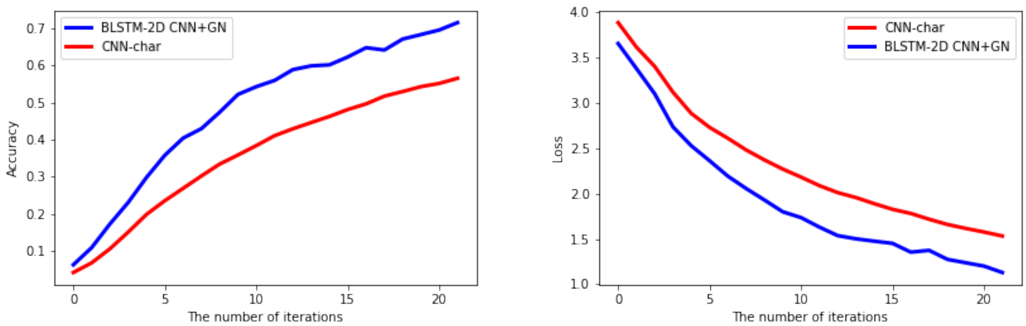
Models	CCAT50	IMDb62	Blog50	Twitter
CNN-char (Ruder et al., 2016)	76.77	91.01	59.70	84.29
CNN-word (Ruder et al., 2016)	77.03	92.30	62.40	82.50
BertAA (Fabien et al., 2020)	78.90	93.00	64.40	62.50
Our method	83.42	93.72	60.67	87.76

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



(a) Accuracy (b) Loss

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



(a) Accuracy (b) Loss

Figure 3: The accuracy and convergence curve over iterations on Twitters dataset.

Transactions on Biometrics, Behavior, and Identity Science.

Shlomo Argamon and Shlomo Levitan. 2005. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*, pages 4–7.

Hosein Azarbonyad, Mostafa Dehghani, Maarten Marx, and Jaap Kamps. 2015. Time-aware authorship attribution for short text streams. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 727–730.

Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks. *CoRR*, abs/1506.04891.

Janek Bevendorff, Matthias Hagen, Benno Stein, and Martin Potthast. 2019. Bias analysis and mitigation in the evaluation of authorship verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6301–6306.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Erik Cambria and Bebo White. 2014. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Steven HH Ding, Benjamin CM Fung, Farkhund Iqbal, and William K Cheung. 2017. Learning stylometric representations for authorship analysis. *IEEE transactions on cybernetics*, 49(1):107–121.

Maël Fabien, Esaú Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. 2020. Bertaa: Bert fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137.

Tomáš Foltỳnek, Norman Meuschke, and Bela Gipp. 2019. Academic plagiarism detection: a systematic literature review. *ACM Computing Surveys (CSUR)*, 52(6):1–42.

Helena Gómez-Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov, and David Pinto. 2018. Document embeddings learned on various types of n-grams for cross-topic authorship attribution. *Computing*, 100(7):741–756.

Bhumika Gupta, Monika Negi, Kanika Vishwakarma, Goldi Rawat, Priyanka Badhani, and B Tech. 2017. Study of twitter sentiment analysis using machine learning algorithms on python. *International Journal of Computer Applications*, 165(9):29–34.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Farkhund Iqbal, Hamad Binsalleeh, Benjamin CM Fung, and Mourad Debbabi. 2013. A unified data mining solution for authorship analysis in anonymous textual communications. *Information Sciences*, 231:98–112.

Fereshteh Jafariakinabad, Sansiri Tarnpradab, and Kien A Hua. 2019. Syntactic recurrent neural network for authorship attribution. *arXiv preprint arXiv:1902.09723*.

Patrick Juola. 2007. Future trends in authorship attribution. In *IFIP International Conference on Digital Forensics*, pages 119–132. Springer.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*.

AdamJB KingaD. 2015. A methodforstochasticoptimization. *Anon. InternationalConferenceon Learning Representations. SanDego: ICLR*.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Robert Layton, Paul A Watters, and Richard Dazeley. 2015. Authorship analysis of aliases: Does topic influence accuracy? *Natural Language Engineering*, 21(4):497–518.

A Pastor López-Monroy, Fabio A Gonzalez, and Thamar Solorio. 2020. Early author profiling on twitter using profile features with multi-resolution. *Expert Systems with Applications*, 140:112909.

Kim Luyckx and Walter Daelemans. 2011. The effect of author set size and data size in authorship attribution. *Literary and linguistic Computing*, 26(1):35–55.

David Madigan, Alexander Genkin, David D Lewis, and Dmitriy Fradkin. 2005. Bayesian multinomial logistic regression for author identification. In *AIP conference proceedings*, volume 803, pages 509–516. American Institute of Physics.

Ilia Markov, Efstathios Stamatatos, and Grigori Sidorov. 2017. Improving cross-topic authorship attribution: The role of pre-processing. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 289–302. Springer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Abiodun Modupe, Oludayo O Olugbara, and Sunday O Ojo. 2014. Filtering of mobile short messaging service communication using latent dirichlet allocation with social network analysis. In *Transactions on Engineering Technologies*, pages 671–686. Springer.

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

Smita Nirkhi, RV Dharaskar, and VM Thakare. 2016. Authorship verification of online messages for forensic investigation. *Procedia Computer Science*, 78:640–645.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2814–2819.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Spyridon Plakias and Efstathios Stamatatos. 2008. Tensor space models for authorship identification. In *Hellenic Conference on Artificial Intelligence*, pages 239–249. Springer.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2016. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing*, 174:50–59.

Juan-Pablo Posadas-Durán, Helena Gómez-Adorno, Grigori Sidorov, Ildar Batyrshin, David Pinto, and Liliana Chanona-Hernández. 2017. Application of the distributed document representation in the authorship attribution task for small corpora. *Soft Computing*, 21(3):627–639.

Hoshiladevi Ramnial, Shireen Panchoo, and Sameerchand Pudaruth. 2016. Authorship attribution using stylometry and machine learning techniques. In *Intelligent Systems Technologies and Applications*, pages 113–125. Springer.

Anderson Rocha, Walter J Scheirer, Christopher W Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne RB Carvalho, and Efstathios Stamatatos. 2016. Authorship attribution for social media forensics. *IEEE transactions on information forensics and security*, 12(1):5–33.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102.

Yunita Sari, Mark Stevenson, and Andreas Vlachos. 2018. Topic or style? exploring the most useful features for authorship attribution. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 343–353.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. 2017. Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 267–273.

Raheem Sarwar, Chenyun Yu, Ninad Tungare, Kanatip Chitavisutthivong, Sukrit Sriratanawilai, Yaohai Xu, Dickson Chow, Thanawin Rakthanmanon, and Sarana Nutanong. 2018. An effective and scalable framework for authorship attribution query processing. *IEEE Access*, 6:50030–50048.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2011. Authorship attribution with latent dirichlet allocation. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 181–189.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship attribution with topic models. *Computational Linguistics*, 40(2):269–310.

Prasha Shrestha, Sebastian Sierra, Fabio A González, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.

Efstathios Stamatatos and Moshe Koppel. 2011. Plagiarism and authorship analysis: introduction to the special issue. *Language Resources and Evaluation*, 45(1):1–4.

Justin Anthony Stover, Yaron Winter, Moshe Koppel, and Mike Kestemont. 2016. Computational authorship verification method attributes a new work to a major 2nd century a frican author. *Journal of the Association for Information Science and Technology*, 67(1):239–242.

Jianwen Sun, Zongkai Yang, Sanya Liu, and Pei Wang. 2012. Applying stylometric analysis techniques to counter anonymity in cyberspace. *Journal of Networks*, 7(2):259.

Leo Wanner et al. 2017. On the relevance of syntactic and discourse features for author profiling and identification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 681–687.

Haiyan Wu, Zhiqiang Zhang, and Qingfeng Wu. 2021. Exploring syntactic and semantic features for authorship attribution. *Applied Soft Computing*, 111:107815.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

Richong Zhang, Zhiyuan Hu, Hongyu Guo, and Yongyi Mao. 2018. Syntax encoding with application in authorship attribution. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2742–2753.

Zhenjie Zhao, Andrew Cattle, Evangelos Papalexakis, and Xiaojuan Ma. 2019. Embedding lexical features via tensor decomposition for small sample humor recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Deyu Zhou, Yang Yang, and Yulan He. 2018. Relevant emotion ranking from text constrained with emotion relationships. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 561–571.