Syntax Encoding with Application in Authorship Attribution

Richong Zhang
Beihang University, Beijing, China
zhangrc@act.buaa.edu.cn

Zhiyuan Hu
Beijing University of Chemical Technology
zhiyuan.hu.bj@gmail.com

Hongyu Guo
National Research Council Canada
hongyu.guo@nrc-cnrc.gc.ca

Yongyi Mao
University of Ottawa, Ottawa, Ontario
ymao@uottawa.ca

Abstract

We propose a novel strategy to encode the syntax parse tree of sentence into a learnable distributed representation. The proposed syntax encoding scheme is provably information-lossless. In specific, an embedding vector is constructed for each word in the sentence, encoding the path in the syntax tree corresponding to the word. The one-to-one correspondence between these "syntax-embedding" vectors and the words (hence their embedding vectors) in the sentence makes it easy to integrate such a representation with all word-level NLP models. We empirically show the benefits of the syntax embeddings on the Authorship Attribution domain, where our approach improves upon the prior art and achieves new performance records on five benchmarking data sets.

1 Introduction

Syntactic parse information plays an essential role in interpreting natural languages because natural language sentences are typically structured in a linguistic grammar. As such, in many NLP applications it is desirable to extract syntactic features from text or sentences, for which there exist a rich body of literature (Baayen et al., 1996; Hirst, 2007; Massung et al., 2013; Wang et al., 2015; Socher et al., 2011; Zhu et al., 2015b; Tai et al., 2015; Zhu et al., 2015a)

To date, existing approaches to exploit syntactic information can be categorized into two categories. The first category may be regarded as "syntactic feature engineering". In such approaches, certain properties or statistics are extracted from the syntax parse tree of a sentence as the syntactical feature. For example, the extracted feature may include the depths of the tree, frequency of certain structural patterns in the tree and so on (Massung et al., 2013; Wang et al., 2015). The advantage of such a method is that the extracted feature can be used for any kind of classifier, if the feature is deemed relevant to the classification task. The limitation of such an approach is however that rich structural information contained in the syntax tree is lost in the feature extraction process. Additionally, with such a strategy, the model designer is often required to design syntactic feature extractors specific for his tasks.

The second category may be regarded as "syntax-assisted sentence coding". This category of approaches build upon neural network models. Examples of such approaches include Tree-LSTM (Tai et al., 2015; Zhu et al., 2015b) and Recursive Neural Networks (Socher et al., 2011), in which the networks are structured according to the syntax tree of the input sentence. The network, after being trained, is capable of encoding a sequence of word embeddings, in a bottom-up manner, to a vector representing the entire sentence. It is worth noting that with these approaches, the encoded feature vector, although containing syntactical information, mainly serves as a *semantic* representation of the input sentence, and the syntactic information exploited therein primarily serves to assist the semantic representation. Additionally, such an approach is not flexible enough to be integrated with another popular class of NLP models, CNN.

One motivation of this work is to develop a generic representation of the parse structure of sentences. Ideally, we would like the representation to maximally preserve the syntactical information and can be integrated easily with any neural network NLP models. This latter requirement is particularly desirable, in light of the competitive performance of CNN and their advantages in training efficiency.

Another motivation of this work is the application of Authorship Attribution (AA) (Hirst, 2007; Stamatatos, 2009; Ouamour and Sayoud, 2012; Stamatatos, 2011; De Vel et al., 2001; Rocha et al.,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2742–2753
Brussels, Belgium, October 31 - November 4, 2018.

2017b; Binongo, 2003; Sohn et al., 2015; Shrestha et al., 2017b). In this application, one is to extract information from a document so as to infer the document's author, from a given list of candidates. There have been a variety of use cases of AA in practice, which include, amongst many others, plagiarism detection (Stamatatos and Koppel, 2011), forensics (Rocha et al., 2017a), suicidal note verification (Chaski, 2005), and intelligent question answering (Stamatatos, 2006). In this application, previous works have mostly focused on building a classifier based on content-level features. The hypothesis underlying our approach to AA in this paper is that syntactic information is a useful additional feature that can characterize an author. This is not only because that syntactical information complements the document content in language understanding, it is also due to the fact that different authors may construct sentences with varying distributions of syntactical structures. In a sense, the "syntactical pattern" of an author may characterize in part the "writing style" of the author, which is independent of the content he writes and the semantics of the document.

To that end, we propose a novel strategy to encode the syntax parse tree of sentence into a learnable distributed representation. Briefly, in this representation, an embedding vector is constructed for each word in the sentence, encoding the path in the syntax tree corresponding to the word. The one-to-one correspondence between these "syntax-embedding" vectors and the words (hence their embedding vectors) in the sentence makes it easy to integrate such a representation with all word-level NLP models.

The proposed syntax encoding scheme has a remarkable property, namely that it is provably information-lossless, as long as the syntax embedding space has adequate dimension. This property also distinguishes the proposed scheme from other distributed representations of syntax.

We apply the proposed syntax encoding approach to the state-of-the-art CNN-based AA model and evaluate its performance over five benchmarking datasets. Experimental study verifies the effectiveness of the proposed syntax encoding scheme. In fact, over all five datasets, syntax-augmented CNN model demonstrates new state-of-the-art performance.

2 Related Work

The aim of the related work here is twofold: syntax encoding and authorship attribution.

2.1 Syntax Encoding

Including syntactic parse information to benefit NLP models have been actively investigated in decades. Syntactic feature engineering refers to efforts to statically extract domain specific features from syntax parse tree of the given text (Massung et al., 2013; Wang et al., 2015). Recent attempts also include leveraging syntactic parse tree structure to recursively generate sentence representations bottom-up (Socher et al., 2011; Zhu et al., 2015b; Tai et al., 2015; Zhu et al., 2015a).

Both the above two categories of methods have severe limitations. The former parse representation typically fails to encode the parse tree structure, and the latter is constrained by the tree structures favored by the parser. Furthermore, the recent distributed word embedding techniques, such as Glove (Pennington et al., 2014) and W2V (Mikolov et al., 2013), have been shown to encode limited syntax knowledge of the given corpus (Andreas and Klein, 2014). This shortcoming has also promoted recent research on creating syntax-aware word embedding, which enhances the distributed embedding vectors with position information of the word within its surrounding context (Cheng and Kartsaklis, 2015), which again encodes limited syntax information.

Our syntax embedding method overcomes the above mentioned limitations, as previously discussed in Section 1.

2.2 Authorship Attribution

Various AA models make use of SVM classifiers on some carefully engineered lexical or syntactic feature. These works include (Pillay and Solorio, 2010; Varela et al., 2011; Segarra et al., 2013; Seker et al., 2013; Seroussi et al., 2010; Castillo et al., 2015). In many of these models and among many others (e.g., (Koppel et al., 2009; Peng et al., 2003; Apté et al., 1994)), character n-grams are chosen as an important feature.

Recently, researchers have relied on CNN to extract features automatically. In (Ferracane et al., 2017), for instance, a CNN is used on 2-gram embeddings to learn the discourse information for the AA task. In (Shrestha et al., 2017b), a CNN is employed on n-gram embeddings to learn richer

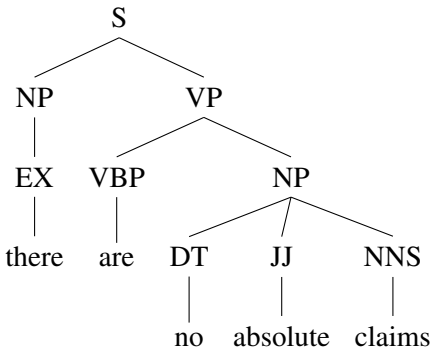

Figure 1: Syntax Tree Example

Words	Syntax Path
there	S→NP→EX
are	S→VP→VBP
no	S→VP→NP→DT
absolute	S→VP→NP→JJ
claims	S→VP→NP→NNS

Table 1: Syntax Path Example

features. A study by (Tang, 2013) considers using SVM as an activation layer for the CNN, replacing the soft-max layer. Nevertheless, a small performance advantage is demonstrated by the method at a high computation cost.

Our proposed model, apart from syntax encoding, is the most close to the CNN architecture of (Shrestha et al., 2017b) where n-grams are used to extract global content information representing content features. The detailed model architecture will be given in a later section.

3 Syntax Encoding

The syntactical structure of a given sentence can be uniquely represented by a tree, which we refer to as the *syntax tree*. An example of such a syntax tree is given in Figure 1. As seen in the example, a syntax tree has labeled nodes. Specifically, the label of each node is a "syntax token", such as `S`, `NP`, `VP`, etc., representing the grammatical property of the word sequences covered by tree branches underneath the node. For example, the root of the tree is always labeled by `S` ("sentence"), and the branches underneath the tree cover the entire sentence. On the other hand, the labels for terminals or leafs of the tree, such as `EX`, `VBP`, `JJ` etc., correspond to "part-of-speech" tags of each word in this sentence. We will denote by $\mathcal{T}$ the set of all syntax tokens.

Given this syntactic tree structure for a sentence s, each word w in sentence s has a unique path in the tree leaving the root and arriving at a terminal. Such a "syntax path" for the word w can then be represented by a sequence of node labels along the path. Some examples of syntax paths are given in Table 1. The following lemma is easy to verify.

Lemma 1 *Let a sentence s be written as a sequence of words $(w_1, w_2, \ldots, w_n)$. For each word position $i = 1, 2, \ldots, n$, let $r(w_i)$ denote the syntax path of word w_i. Let $\mathcal{R} := \{(i, r(w_i)) : i = 1, 2, \ldots, n\}$ be an (unordered) set containing precisely all syntax paths for the words in s. Then the syntax tree of s can be uniquely recovered by $\mathcal{R}$.*

In the lemma, we note that $\mathcal{R}$ is an unordered set. That is, regardless of the ordering of the paths in $\mathcal{R}$, one can always recover the syntax tree from $\mathcal{R}$.

Let $r(w)$ be the syntax path of a word w in the sentence s of interest. Specifically, $r(w)$ can be written as the sequence $(t_1, t_2, \ldots, t_L)$, where L is the number of nodes in the path r, and each t_i is a syntax token.

Let the Euclidean space $\mathbb{R}^K$ be the embedding space which we will use to encode syntax. We now describe a method that encodes the path $r(w)$ into a vector $\overline{r(w)} \in \mathbb{R}^K$.

Let $L_{\max}$ be maximum depth of the syntax trees in the corpus. For each $i = 1, 2, \ldots, L_{\max}$, let $\overline{p}_i$ be a vector in $\mathbb{R}^K$ serving as the embedding for integer i. Here integer i is meant to indicate the location of a token in a syntax path. For each $t \in \mathcal{T}$, let $\overline{t}$ also be a vector in $\mathbb{R}^K$, serving as the embedding for syntax token t. Let vector $\overline{r(w)} \in \mathbb{R}^K$, the embedding of path $r(w)$, be defined by

$$\overline{r(w)} := \sum_{t_j \in r(w)} \overline{t}_j \circ \overline{p}_j \qquad (1)$$

where $\circ$ is the element-wise product operation. For example in Table 1, the syntax path for word "there" will have embedding $\overline{\text{NP}} \circ \overline{p}_1 + \overline{\text{EX}} \circ \overline{p}_2$; the syntax path for word "no" will have embedding $\overline{\text{VP}} \circ \overline{p}_1 + \overline{\text{NP}} \circ \overline{p}_2 + \overline{\text{DT}} \circ \overline{p}_3$. Note that when embedding a syntax path, the beginning token S is removed from the path since it exists in every path.

Lemma 2 *There exists a random assignment of the vectors $\{\overline{t} : t \in \mathcal{T}\}$ and $\{\overline{p}_i : i = 1, 2, \ldots, L\}$ such that the syntax path $r(w)$ can be recovered from its embedding $\overline{r(w)}$ almost surely for sufficiently large K.*

This lemma (proof given in Appendix) suggests that as long as we choose a sufficiently large embedding dimension K, the above introduced encoding for syntax paths is essentially lossless. Thus, for a given n-word sentence $s = (w_1, w_2, \ldots, w_n)$, if we collect the embedding vectors $\overrightarrow{r(w_1)}, \overrightarrow{r(w_2)}, \ldots, \overrightarrow{r(w_n)}$ of all syntax paths and list them as the columns of a $K \times n$ matrix, then by Lemmas 1 and 2, the syntax tree of s can be recovered from the matrix. Such a matrix is then an information lossless encoding of the syntax tree.

We note however that in practice, when the tokens embeddings and the position (integer) embeddings are learned, there is no longer a guaranty that a syntax path can be recovered from its embedding. This is particularly the case for supervised tasks. During the training for such tasks, the information irrelevant to the training objective is necessarily "squeezed out", and the representations of those syntax paths that provide no distinguishing features are "pulled closer". This will cause these paths non-distinguishable (and hence non-recoverable) from their embeddings. This is also the reason that in practice there is no need to have very large embedding dimension K.

Nonetheless, since different supervised tasks may have distinct training objectives, a "lossy" syntax encoding suitable for one task may prove ineffective for other tasks. Thus it is still essential to adopt an information-lossless encoding framework, as we propose in this paper, that is universally applicable.

Next, we will discuss the application of our syntax encoding approach to AA models.

4 Authorship Attribution Model

4.1 Problem Definition

The objective of Authorship Attribution is to develop a classifier that predicts the authors of unseen documents based on a given set of documents and their corresponding authors. Despite the previous successes in solving the AA problem discussed in a previous section, we argue that the syntactic parse information in an author's writing can characterize in part the "writing style" of the author. Specifically, even when writing the same content, two authors may prefer using different syntax structure in constructing their sentences. Consider the following two sentences.

A1: Take a left at the end of the street, you will see the house.

A2: You will see the house, if you go down to the end of the street and take a left.

The two sentences have the same meaning and yet two different authors may favor different ones over the other. This provides opportunity for leveraging such syntactical information as an additional feature, beyond the lexical and semantic features, to distinguish the two authors.

Detecting the syntactical differences among authors suits well the application scope of the proposed syntax encoding scheme. As such, in this work, we will use the AA problem as an test bed to examine the effectiveness of our scheme.

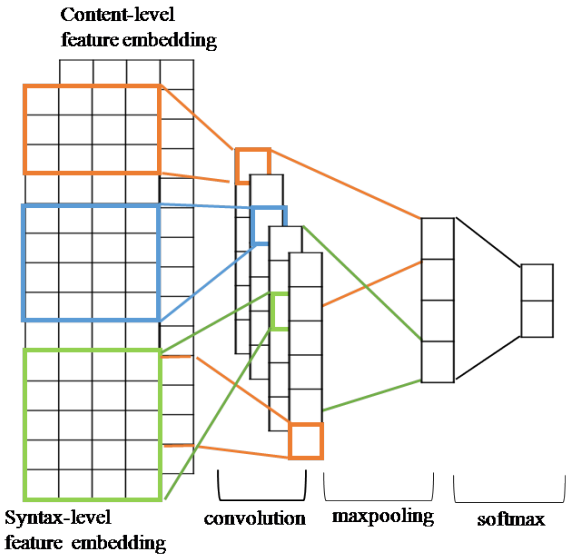

Figure 2: Single Layer CNN Architecture

4.2 Syntax-augmented CNN Model

The overall architecture of our model is shown in Figure 2. Our model takes advantage of the expressive power of convolutional neural networks(CNNs) to learn the embeddings for both of the content-level features and the syntax-level features. The model consists of 5 types of layers: *Syntax-level feature embedding, Content-level feature embedding, Convolution, Max-pooling* and *Softmax*.

4.2.1 Overall Structure

The model takes both the context-level features and the syntax-level features as the input.

As the character n-gram features have been used successfully in both of the text classification

tasks (Kim, 2014; Chen et al., 2017) and the AA tasks (Shrestha et al., 2017b; Sapkota et al., 2015; Shrestha et al., 2017a; Sari et al., 2017; Ruder et al., 2016), in this study, we use the character n-grams to represent the content-level features.

We align in tandem the embedding vectors of consecutive n-grams of a document to form a content-level embedding matrix, and the consecutive syntax-path embeddings of each word to form the syntax-level embedding matrix. These embedding matrices are individually passed to 2 parallel CNNs having different filter lengths. The convolutional layer outputs feature maps via convolutional filtering. The max-pool layer is then applied across different feature maps to form the content-level and syntax-level representation of a document respectively. These two representations are then concatenated to form the final feature vector characterizing the author of the document. Finally, this feature vector is passed to a learnable softmax layer. The number of outputs of the softmax layer is the number of authors, which the i^{th} output is the probability that the document written by author i.

4.2.2 Convolution and Max-pool

In the convolutional layer, we capture local contextual information using kernels, which combine the vectors within it's window as it slides over the embedding matrix. A linear transformation processes the output of the kernel.

We present a clean description for this operation. Let $E \in \mathbb{R}^{k \times d}$ be the embedding matrix containing embeddings $e_{1:k}$, where $e_i \in \mathbb{R}^d$ is the i-th embedding in E, l is the number of filters and w is the filter length or window size. Let $F \in \mathbb{R}^{w \times d}$ be the filter matrix and a bias vector b. We define the vector $g_i \in \mathbb{R}^{w \times d}$ as the concatenation of w embeddings in the i-th window, where

$$g_i = e_{i-w+1:i} \quad 1 \leq i \leq k + w - 1 \tag{2}$$

The result of filter F across embedding matrix E outputs a feature map $f_j \in \mathbb{R}^d$ for the j-th kernel where the i-th value of f_j is computed as

$$f_{ji} = F \otimes g_i + b \tag{3}$$

where $\otimes$ denotes the convolution operator. Let $f \in \mathbb{R}^{l \times d}$ denote the concatenation of the l feature maps for compactness. Further to this, we apply a ReLU activation function before passing the output to the max-pool layer. Given l filters of different dimensions, we obtain l feature maps. Each feature map encodes different types of abstract contextual information globally for the matrix E. To obtain the most relevant features from each dimension of the feature maps, we max-over feature map dimensions expressed as:

$$m_i = max(f(\cdot, i)), \quad 1 \leq i \leq d \tag{4}$$

where m_i represents the maximum value for dimension i across the l feature maps. The feature vector m from the max-pool operation is the concatenation of all m_i. For our 2 CNN's we obtain a content-level vector representation and a syntax-level representation for an author's text. We denote $\bar{m} \in \mathbb{R}^t$ as the concatenation of these 2 vectors representing an author's style. We obtain the confidence of an author's attribution to a text by feeding this final vector to a softmax layer. In the next section, we give a brief description of the softmax layer.

4.2.3 Softmax Layer

Given a text input x and CNN parameters θ, this layer outputs a score for all n authors. The output of the softmax layer is a vector with dimension equal to the number of author labels. To compute the confidence of author's attribution, the author's style representation $\bar{m}$ is transformed by a transformation matrix $W \in \mathbb{R}^{n \times t}$.

$$o = W\bar{m} \tag{5}$$

where the i-th component of o corresponds to the confidence score of author i. A softmax operation is called on o to compute the conditional probability for each dimension. This is calculated by:

$$p(i|x, \theta) = \frac{e^{o_i}}{\sum_{j=1}^{n} e^{o_j}} \tag{6}$$

To train our model, the log-likelihood of the probability should be maximized. To predict the attribution of authors, the label with the highest probability is selected. For the optimization of our model, we use SGD algorithm to solve the optimization problem.

We denote our syntax augmented CNN model as Syntax-CNN.

5 Experimental Studies

We first empirically show the predictive performance of the Syntax-CNN strategy, which establishes new state-of-the-art accuracy for several

Description	CCAT10	CCAT50	IMDB62	blogs10	blogs50
Avg # of words per doc	580	584	345	108.6	117.1
Avg # of chars per doc	3,089	3,010	1742	502.3	541.5
Docs per author	100	100	1000	2353.4	1470.1
Max # of chars	8,716	8,716	11617	30712	30712
Min # of chars	483	345	82	3	3

Table 2: Statistics of Datasets

benchmarking data sets. We then provide ablation studies, aiming at better understanding the contributions of the syntax embeddings to the Syntax-CNN method.

5.1 Experimental Setup

5.1.1 Datasets

We test the Syntax-CNN approach on several benchmarking datasets. Summary statistics of the datasets are in Table 2.

CCAT10: 100 newswire stories, written by 10 authors, in English taken from Reuters Corpus Volume 1 (RCV1) (Stamatatos, 2008).

CCAT50: Same as CCAT10 but with 100 news article written by 50 authors.

IMDB62: 62,000 movie reviews and 17,550 message posts from 62 prolific authors obtained from Internet Movie Database(IMDb) (Seroussi et al., 2010).

Blogs10: The original data set contains 681,288 blog posts by 19,320 bloggers for blogger.com. Posts written by the top ten bloggers are selected for the Blogs10 data set (Schler et al., 2006).

Blogs50: Same as Blogs50 but with the posts written by the top 50 bloggers.

5.1.2 Compared Prior Art

We compare our method with various baseline approaches, which represent the current art in the AA problem. They include SCAP (Frantzeskou et al., 2007), SVM with 2,500 most frequent 3-grams (Plakias and Stamatatos, 2008), SVM with bag of local histogram (Escalante et al., 2011), Imposters (Koppel et al., 2011), LDAH-S (Seroussi et al., 2011),SVM with affix and punctuation 3-grams (Sapkota et al., 2015), CNN-char (Ruder et al., 2016), Continuous n-gram representation (Sari et al., 2017), and N-gram CNN (Shrestha et al., 2017a). Except the last two methods, all the results reported in this paper were obtained from their respective papers. In all our experiments, we partitioned the datasets into train/dev/test in the same way as are used in the literature in order for fair comparison.

5.1.3 Hyperparameters and Training

Our experimental setup follows that of the current state-of-the-art AA method (Shrestha et al., 2017b). In specific, the networks are trained using mini-batches with size of either 16 (for IMDB62,Blogs10 and Blogs50) or 32 (for CCAT10 and CCAT50). We use 3-gram with embedding size of 300 for character, and embedding size of 60 for syntax vector; these embeddings are randomly initialized. We apply Adagrad (Duchi et al., 2011) with initial learning rate of 1e-4. For the CNN, we use filter sizes of 3, 4, and 5 with 50 feature maps for syntax embedding; 500 dimensional character embedding for CCAT, IMDB, and Blogs50, and 200 for Blog10. We train for at most 300 epochs, with a dropout rate of 0.25. We deploy early stop strategy for the training using a validation data set, which contain 10% of the randomly selected samples from the training set. We stop the training when the validation loss goes up.

In our experiments, we use the Stanford CoreNLP parser. For documents contain more than one sentences, each sentence is parsed separately. The syntax encoding for each word is done according to its syntax path in syntax tree containing the word. The syntax embeddings of all words in the document form the input matrix to the CNN responsible for extracting syntactical features.

5.2 Predictive Performance

Table 3 presents the accuracy obtained by various testing methods on the five benchmarking datasets, where the best result of each data set is highlighted in bold.

5.2.1 Accuracy Obtained by Syntax-CNN

Results in Table 3 indicate that the Syntax-CNN outperformed all the testing methods on the five benchmarking datasets, beating the current best records on these testing AA tasks. Our further

Model	CCAT10	CCAT50	IMDB62	Blogs10	Blogs50
SCAP(2007)	#	#	94.8	48.6	41.6
SVM with most frequent 3-grams(2008)	80.8	67	81.4	#	#
SVM with bag of local histogram(2011)	86.4	#	#	#	#
Imposters(2011)	#	#	76.9	35.4	22.6
LDAH-S(2011)	#	#	72.0	52.5	18.3
SVM with affix+punctuation 3-grams(2015)	78.8	69.3	#	#	#
CNN-char(2016)	#	#	91.7	61.2	49.4
Continuous n-gram representations(2017)	74.8	72.6	95.12	61.34	52.82
N-gram CNN(2017)	86.8	76.5	95.21	63.74	53.09
Syntax	22.8	10.08	83.48	48.64	42.91
Syntax-CNN	**88.20**	**81.00**	**96.16**	**64.10**	**56.73**

Table 3: Accuracy obtained by the testing methods; best result for each data set is in **bold**.

Quality metrics	CCAT10	CCAT50	IMDB62	Blogs10	Blogs50
normal	18.58%	18.79%	55.81%	82.15%	82.13%
hard	5.96%	6.3%	19.05%	8.61%	8.9%
very hard	75.46%	74.91%	25.14%	9.24%	8.97%

Table 4: Syntax correctness as measured by the Heminway Editor tool

analysis also shows that, the predictive improvement on some datasets is large. For example, against the CCAT50 and Blogs50 datasets, the relative error reductions over the current state-of-the-art result are 5.5% and 3.64% , respectively.

5.2.2 Contribution of Syntax Encoding

We also include the accuracy obtained by using the syntax embeddings alone on the second last row in Table 3. These results indicate that using only the syntax style embeddings achieved very low accuracy on some of the datasets, for example, with only 10.08% on the CCAT50 dataset. As shown in Table 4, the CCAT50 is, indeed, one of hardest datasets, as judged by the Hemingway Editor tool [1], which aims to measure the syntax correction of a given piece of text. Nevertheless, such syntax style embeddings can bring significant accuracy gain to the CNN strategies. As shown in Table 3, for the CCAT50 dataset, a relative error reduction of 5.5% (which represents the largest error reduction of the five testing datasets) was achieved by Syntax-CNN over the best performed CNN model, i.e., the N-gram CNN. Similar behavior can also been observed for the CCAT10 dataset as shown in Tables 4 and 3. These results suggest that the Syntax-CNN may favor sentences with syntax difficulties.

[1] http://www.hemingwayapp.com

We also provide, in Table 5, further information about the percentage of classifications corrected or mislabeled when enabling the syntax style embeddings in the Syntax-CNN method on both the CCAT10 and CCAT50 datasets. Table 5 clearly indicates that with the syntax style embeddings deployed, the Syntax-CNN was able to, respectively for the CCAT10 and CCAT50 datasets, correct 43.48% and 37.7% of the testing examples which were mislabeled when the syntax embeddings was disabled in the Syntax-CNN.

data set	CCAT10	CCAT50
wrong-to-correct	43.48%	37.7%
correct-to-wrong	4.87%	6.1%

Table 5: Percentage of classifications corrected (wrong-to-correct) or mislabeled (correct-to-wrong) when using the syntax style embeddings.

5.2.3 Syntactic Style Examples

The contributions of the syntax information is further justified by examining documents whose classifications are corrected when enabling the syntax embedding of the Syntax-CNN. For example, the following two excerpts, extracted from documents written by two different authors, share great semantic similarity. But they show a difference in

terms of syntax information contained. In particular, the former is constructed by deep parse trees, but the later has a succinct parse tree structure.

> author1:
>
> China's long-delayed bid to enter the World Trade Organization will fall under the spotlight when the world's richest nations meet to discuss its application this week. United States says Beijing must comply with a "road map" to open its markets and eliminate trade and non-trade barriers before it can win U.S. support for its entry.
>
> author2:
>
> China must make real changes to its economy if it wants to join the World Trade Organization and should replace anti-U.S. rhetoric with international cooperation.

Using the two documents containing these two excerpts, we perform a separate small experiment. When we remove syntax encoding and its corresponding feature from Syntax-CNN, the model fails to classify the authors of the two documents. But when syntax encoding is included, the model can distinguish the two authors. Additionally, on the syntax encoding side, when we only keep the syntax path encoding for these two excerpts in their respective documents and remove all other sentences from the two documents, Syntax-CNN is still able to classify the two authors correctly.

This suggests that syntactic information is indeed useful for authorship attribution, and Syntax-CNN is able to extract such information effectively.

5.3 Model Behaviors

This section aims to further evaluate the behaviors of the Syntax-CNN model.

5.3.1 Sensitivity to Syntax Embedding Dimension

To understand the impact of the embedding size of the syntax tree in the Syntax-CNN method, we conduct further experiments on the Blogs50 dataset. We vary the dimension of the syntax tree embedding, from 5 up to 150. The experimental results are reported in Table 6.

Table 6 shows that Syntax-CNN is robust to the dimension size of syntax style embedding. The lowest accuracy was obtained by Syntax-CNN with an embedding dimension of 150. Although we have noted earlier that using a higher syntax embedding dimension allows the syntactical information to be better preserved. A downside of such a choice is however the risk of overfitting, which is expectedly the cause of the degradation of the performance at dimension 150. Note that even in this case, the performance of Syntax-CNN is still higher than the current state-of-the-art accuracy of 53.09% achieved by the N-gram CNN.

dimension	Syntax-CNN
5	54.96
30	55.86
60	56.82
100	56.42
150	53.34

Table 6: Accuracy obtained by Syntax-CNN on the Blogs50 while varying the syntax style embedding dimension.

data set	with	without
Blogs50	56.73	55.05
Blogs10	64.10	62.70

Table 7: Accuracy obtained by Syntax-CNN with and without position vector.

5.3.2 Impact of Position Vector

We further evaluate the impact of the position embedding vectors (i.e., the $\overline{p}_j$ vectors in Section 3) in syntax encoding. We conduct experiments on both the Blogs10 and Blogs50 datasets with and without the position embedding. In particular, by "without position", we mean that the element-wise multiplication with the $\overline{p}_j$ is removed from equation (1) in Section 3. The results are reported in Table 7.

Table 7 indicates that the position vectors play an important role in capturing syntactical information. In fact, it is possible to prove that without multiplying with the position vectors, in general the syntax tree is no longer recoverable from the encoding $\overline{r(w)}$ of syntax path $r(w)$. Additionally, the results in Table 7 suggest that depth information in the syntax tree is an important feature for distinguishing the writing styles of the authors.

6 Conclusion

We propose a novel strategy to encode a syntax tree into a learnable distributed representation.

This representation can be easily integrated into any NLP neural network model and entails no loss of information. Using this representation as an additional input, we extend the CNN architecture introduced in (Shrestha et al., 2017b) for the authorship attribution problem. Experimental evaluation of the extended model on the standard author attribution datasets demonstrates the effectiveness of the proposed syntax encoding approach. Record-breaking performances are obtained.

Acknowledgments

This work is supported partly by China 973 program (2015CB358700), by the National Natural Science Foundation of China (61772059,61421003), and by the Beijing Advanced Innovation Center for Big Data and Brain Computing and State Key Laboratory of Software Development Environment.

Appendix: Proof of Lemma 2

We first establish some elementary results.

Lemma 3 *Suppose that random variables X, Y, Z, U are independent standard normal random variables. Then*

1. $\mathrm{E}(XYZU) = 0$ *and* $\mathrm{VAR}(XYZU) = 1$.
2. $\mathrm{E}(X^2YZ) = 0$ *and* $\mathrm{VAR}(X^2YZ) = 3$.
3. $\mathrm{E}(X^2Y^2) = 1$ *and* $\mathrm{VAR}(X^2Y^2) = 8$.

These results follow from the independence among (X, Y, Z, U) and the fact that the square of a standard normal random variable is a Chi-Square random variable.

We now construct a random assignment scheme and a recovery scheme.

Random Assignment: Generate each vector in $\{\overline{t} : t \in \mathcal{T}\}$ and in $\{\overline{p}_i : i = 1, 2, \ldots, L_{\max}\}$ by assigning independent values drawn from a standard normal distribution.

Recovery Scheme: Let a path embedding $\overline{r(w)}$ be given. For each token-integer pair $(u, i) \in \mathcal{T} \times \{1, 2, \ldots, L_{\max}\}$, compute vector $\mathbf{a} \in \mathbb{R}^K$ and scalar b by

$$\mathbf{a} := \overline{r(w)} \circ \overline{u} \circ \overline{p}_i, \quad b := \frac{1}{K} \sum_{k=1}^{K} \mathbf{a}[k]$$

where $\mathbf{a}[k]$ is the k^{th} element of $\mathbf{a}$. Choose an arbitrary positive value $\epsilon < 1/2$. If $|b - 1| < \epsilon$, claim the i^{th} token on path $r(w)$ is u.

Now we prove that using this scheme, when the embedding dimension K is sufficiently large, one can recover the path $r(w)$ with probability arbitrarily close to 1.

First

$$\mathbf{a} = \sum_{t_j \in r(w)} \overline{t}_j \circ \overline{p_j} \circ \overline{u} \circ \overline{p}_i$$

Note that this summation contains summing of L K-vectors, which we will re-denote by $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_L$. Also we denote

$$C_l := \frac{1}{K} \sum_{k=1}^{K} \mathbf{c}_l[k].$$

Then we have

$$\mathbf{a} = \sum_{l=1}^{L} \mathbf{c}_l \tag{7}$$

and

$$b = \sum_{l=1}^{L} C_l \tag{8}$$

Each of $\mathbf{c}_l$ terms in Equation (7) corresponds to one of the four cases below.

Case 1: $i = j$ and $u = t_j$. This corresponds to case 3 of Lemma 3, and we have $\mathrm{E}(\mathbf{c}_l[k]) = 1$ and $\mathrm{VAR}(\mathbf{c}[k]) = 8$. It follows that $\mathrm{E}(C_l) = 1$ and $\mathrm{VAR}(C_l) = 8/K$.

Case 2: $i = j$ and $u \neq t_j$. This corresponds to case 2 of Lemma 3, and we have $\mathrm{E}(\mathbf{c}_l[k]) = 0$ and $\mathrm{VAR}(\mathbf{c}[k]) = 3$. It follows that $\mathrm{E}(C_l) = 0$ and $\mathrm{VAR}(C_l) = 3/K$.

Case 3: $i \neq j$ and $u = t_j$. This also corresponds to case 2 of Lemma 3, and we also have $\mathrm{E}(C_l) = 0$ and $\mathrm{VAR}(C_l) = 3/K$.

Case 4: $i \neq j$ and $u \neq t_j$. This corresponds to case 1 of Lemma 3, and we have $\mathrm{E}(\mathbf{c}_l[k]) = 0$ and $\mathrm{VAR}(\mathbf{c}[k]) = 1$. It follows that $\mathrm{E}(C_l) = 0$ and $\mathrm{VAR}(C_l) = 1/K$.

For each of these cases, the distribution of C_l can be made concentrated at its mean when K is made large enough (by invoking either the Weak Law of Large Number or the Central Limit Theorem). Therefore, if there exists a token in the path $r(w)$ that makes Case 1 satisfied, the distribution of b is concentrated at 1, and our recovery scheme will detect, with probability close to 1, the token and its position in the path. On the other hand, if there is no such token, the distribution of b is concentrated at 0, and with probability close to 0 our scheme will make a false detection. □

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *ACL*, pages 822–827.

Chidanand Apté, Fred Damerau, and Sholom M Weiss. 1994. Towards language independent automated learning of text categorization models. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 23–30. Springer-Verlag New York, Inc.

H Baayen, H van Halteren, and F Tweedie. 1996. Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121–132.

José Nilo G Binongo. 2003. Who wrote the 15th book of oz? an application of multivariate analysis to authorship attribution. *Chance*, 16(2):9–17.

Esteban Castillo, Darnes Vilarino, Ofelia Cervantes, and David Pinto. 2015. Author attribution using a graph based representation. In *2015 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 135–142.

Carole E. Chaski. 2005. Who's at the keyboard? authorship attribution in digital evidence investigations. *IJDE*, 4(1).

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *EMNLP*, pages 1531–1542.

Olivier De Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2001. Mining email content for author identification forensics. *ACM Sigmod Record*, 30(4):55–64.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Hugo Jair Escalante, Thamar Solorio, and Manuel Montes-y Gómez. 2011. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 288–298. Association for Computational Linguistics.

Elisa Ferracane, Su Wang, and Raymond Mooney. 2017. Leveraging discourse information effectively for authorship attribution. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 584–593.

Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E Chaski, and Blake Stephen Howald. 2007. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *International Journal of Digital Evidence*, 6(1):1–18.

Graeme Hirst. 2007. Bigrams of syntactic labels for authorship discrimination of short texts. In *Literary and Linguistic Computing*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the Association for Information Science and Technology*, 60(1):9–26.

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

Sean Massung, Chengxiang Zhai, and Julia Hockenmaier. 2013. Structural parse tree features for text representation. In *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, pages 9–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

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

Fuchim Peng, Dale Schuurmans, Vlado Keselj, and Shaojun Wang. 2003. Automated authorship attribution with character level language models. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, pages 267–274.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Sangita R Pillay and Thamar Solorio. 2010. Authorship attribution of web forum posts. In *eCrime Researchers Summit (ECrime), 2010*, pages 1–7. IEEE.

Spyridon Plakias and Efstathios Stamatatos. 2008. Tensor space models for authorship identification. *Artificial Intelligence: Theories, Models and Applications*, pages 239–249.

A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, and E. Stamatatos. 2017a. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33.

Anderson Rocha, Walter J Scheirer, Christopher W Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne RB Carvalho, and Efstathios Stamatatos. 2017b. 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-y-Gmez, 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, 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*, volume 2, pages 267–273.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 6, pages 199–205.

Santiago Segarra, Mark Eisen, and Alejandro Ribeiro. 2013. Authorship attribution using function words adjacency networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5563–5567.

Sadi Evren Seker, Khaled Al-Naami, and Latifur Khan. 2013. Author attribution on streaming data. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 497–503. IEEE.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2010. Collaborative inference of sentiments from texts. In *UMAP*, pages 195–206. Springer.

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, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011*, pages 181–189.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017a. 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*, volume 2, pages 669–674.

Prasha Shrestha, Sebastian Sierra, Fabio A González, Paolo Rosso, Manuel Montes-y Gómez, and Thamar Solorio. 2017b. Convolutional neural networks for authorship attribution of short texts. *EACL 2017*, page 669.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136.

Kyung-Ah Sohn, Tae-Sun Chung, et al. 2015. A graph model based author attribution technique for single-class e-mail classification. In *Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on*, pages 191–196. IEEE.

Efstathios Stamatatos. 2006. Authorship attribution based on feature set subspacing ensembles. 15:823–838.

Efstathios Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management*, 44(2):790–799.

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

Efstathios Stamatatos. 2011. Plagiarism detection using stopword n-grams. *Journal of the Association for Information Science and Technology*, 62(12):2512–2527.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566.

Yichuan Tang. 2013. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*.

Paulo Varela, Edson Justino, and Luiz S Oliveira. 2011. Selecting syntactic attributes for authorship attribution. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 167–172. IEEE.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL*, pages 700–706.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015a. A re-ranking model for dependency parser with recursive convolutional neural network. In *ACL*, pages 1159–1168.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015b. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1604–1612.