A Self-Supervised Representation Learning of Sentence Structure for Authorship Attribution

FERESHTEH JAFARIAKINABAD and KIEN A. HUA, University of Central Florida

The syntactic structure of sentences in a document substantially informs about its authorial writing style. Sentence representation learning has been widely explored in recent years and it has been shown that it improves the generalization of different downstream tasks across many domains. Even though utilizing probing methods in several studies suggests that these learned contextual representations implicitly encode some amount of syntax, explicit syntactic information further improves the performance of deep neural models in the domain of authorship attribution. These observations have motivated us to investigate the explicit representation learning of syntactic structure of sentences. In this article, we propose a self-supervised framework for learning structural representations of sentences. The self-supervised network contains two components; a lexical sub-network and a syntactic sub-network which take the sequence of words and their corresponding structural labels as the input, respectively. Due to the n-to-1 mapping of words to their structural labels, each word will be embedded into a vector representation which mainly carries structural information. We evaluate the learned structural representations of sentences using different probing tasks, and subsequently utilize them in the authorship attribution task. Our experimental results indicate that the structural embeddings significantly improve the classification tasks when concatenated with the existing pre-trained word embeddings.

CCS Concepts: • **Computing methodologies** → *Discourse, dialogue and pragmatics;*

Additional Key Words and Phrases: Sentence representation, sentence structure, neural network, self-supervised learning, authorship attribution

ACM Reference format:
Fereshteh Jafariakinabad and Kien A. Hua. 2022. A Self-Supervised Representation Learning of Sentence Structure for Authorship Attribution. *ACM Trans. Knowl. Discov. Data.* 16, 4, Article 68 (January 2022), 16 pages.
https://doi.org/10.1145/3491203

1 INTRODUCTION

Word embeddings which can capture semantic similarities have been extensively explored in a wide spectrum of **Natural Language Processing** (**NLP**) applications in recent years. Word2Vec [34], FastText [7], and Glove [39] are some examples. Even though distributional word embeddings produce high quality representations, representing longer pieces of text such as sentences and paragraphs is still an open research problem. A sentence embedding is a contextual representation

This work was funded by Crystal Photonics Inc (CPI) under Grant Number 1063271. Any opinions, findings, and conclusion or recommendations expressed in this materials are those of the authors and do not necessarily reflect the views of CPI.
Authors' addresses: F. Jafariakinabad and K. A. Hua, University of Central Florida, 4000 Central Florida Blvd, Orlando, Florida, FL 32816; emails: fereshteh.jafari@knights.ucf.edu, kienhua@cs.ucf.edu.

1556-4681/2022/01-ART68 $15.00
https://doi.org/10.1145/3491203

of a sentence which is often created by transformation of word embeddings through a composition function. There has been a large body of work in the literature which propose different approaches to represent sentences from word embeddings. SkipThought [26], InferSent [11], and Universal Sentence Encoder [9] are well-known examples.

There has been a growing interest in understanding what linguistic knowledge is encoded in deep contextual representation of language. For this purpose, several probing tasks are proposed to understand what these representations are capturing [12, 18, 40, 52]. One of the interesting findings is that despite the existence of explicit syntactic annotations, these learned deep representations encode syntax to some extent [6]. Hewitt et al. provide an evidence that the entire syntax tree is embedded implicitly in deep model's vector geometry. Kuncoro et al. [29] show that LSTMs trained on language modeling objectives capture syntax-sensitive dependencies. Even though deep contextual language models implicitly capture syntactic information of sentences, explicit modeling of syntactic structure of sentences has been shown to further improve the results in different NLP tasks including neural language modeling [15, 46], machine comprehension [32], summarization [49], text generation [5], machine translation [30, 56], authorship attribution [21, 22, 57], and so on. Furthermore, Kuncoro et al. provide evidence that models which have explicit syntactic information result in better performance [29]. Of particular interest, one of the areas where syntactic structure of sentences plays an important role is style-based text classification tasks, including authorship attribution. The syntactic structure of sentences captures the syntactic patterns of sentences adopted by a specific author and reveal how the author structures the sentences in a document.

Inspired by the above observations, our initial work demonstrates that explicit syntactic information of sentences improves the performance of a recurrent neural network classifier in the domain of authorship attribution [21, 22]. We continue this work in this article by investigating if structural representation of sentences can be learned explicitly. In other words, similar to pre-trained word embeddings which mainly capture semantics, can we have pre-trained embeddings which mainly capture syntactic information of words. Such pre-trained word embeddings can be used in conjunction with semantics embeddings in different domains including authorship attribution. For this purpose, we propose a self-supervised framework using a Siamese network [10] to explicitly learn the structural representation of sentences. The Siamese network is consists of two identical components; a lexical sub-network and a syntactic sub-network; which take the sequence of words in the sentence and its corresponding linearized syntax parse tree as the inputs, respectively. This model is trained based on a contrastive loss objective where each pair of vectors (lexical and structural) is close to each other in the embedding space if they belong to an identical sentence (positive pairs), and are far from each other if they belong to two different sentences (negative pairs).

As a result, each word in the sentence is embedded into a vector representation which mainly carries structural information. Due to the n-to-1 mapping of word types to structural labels, the word representation is deduced into structural representations. In other words, semantically different words (e.g., cold, hot, warm) are mapped to similar structural labels (adjective); hence, semantically different words may have similar structural representations. These pre-trained structural word representations can be used as complementary information to their pre-trained semantic embeddings (e.g., FastText and Glove). We use probing tasks proposed by Conneau et al. [12] to investigate the linguistic features learned by such a training. The results indicate that structural embeddings show competitive results compared to the semantic embeddings, and concatenation of structural embeddings with semantic embeddings achieves further improvement. Finally, we investigate the efficiency of the learned structural embeddings of words for the domain of authorship

attribution across four datasets. Our experimental results demonstrate classification improvements when structural embeddings are concatenated with the pre-trained word embeddings.

The remainder of this article is organized as follows: We review the related work in Section 2. We elaborate our proposed self-supervised framework in Section 3. The details of the datasets and experimental configuration are provided and the experimental results reported in Section 4. Finally, we conclude this article in Section 5.

2 RELATED WORK (AUTHORSHIP ATTRIBUTION)

Style-based text classification is dual to topic-based text classification [20, 36] since the features which capture the style of a document are mainly independent of its topic [2, 16]. Writing style is a combination of consistent decisions at different levels of language production including lexical, syntactic, and structural associated to a specific author (or author groups, e.g., female authors or teenage authors) [13]. Style-based text classification was introduced by Argamon-Engelson et al. [2]. The authors used basic stylistic features (the frequency of function words and **part-of-speech** (**POS**) trigrams) to classify news documents based on the corresponding publisher (newspaper or magazine) as well as text genre (editorial or news item). Nowadays, computational stylometry has a wide range of applications in literary science [23, 53], forensics [1, 8, 54], social media analysis [16, 17, 24, 25, 55], and psycholinguistics [35, 38].

Syntactic n-grams are shown to achieve promising results in different stylometric tasks including author profiling [41] and author verification [27]. In particular, Raghahvan et al. investigated the use of syntactic information by proposing a probabilistic context-free grammar for the authorship attribution purpose, and used it as a language model for classification [42]. A combination of lexical and syntactic features has also been shown to enhance the model performance. Sundararajan et al. argue that, although syntax can be helpful for cross-genre authorship attribution, combining syntax and lexical information can further boost the performance for cross-topic attribution and single-domain attribution [51]. Further studies which combine lexical and syntactic features include [28, 45, 48].

With recent advances in deep learning, there exists a large body of work in the literature which employs deep neural networks in the domain of authorship attribution. For instance, Ge et al. used a feed forward neural network language model on an authorship attribution task. The output achieves promising results compared to the n-gram baseline [14]. Bagnall et al. have employed a recurrent neural network with a shared recurrent state which outperforms other proposed methods in PAN 2015 task [3]. Shrestha et al. applied CNN based on character n-gram to identify the authors of tweets. Given that each tweet is short in nature, their approach shows that a sequence of character n-grams as the result of CNN allows the architecture to capture the character-level interactions, which can then be aggregated to learn higher-level patterns for modeling the style [47]. Sari et al. have proposed to use continuous representations for authorship attribution. Unlike the previous work which uses discrete representations, they represent each n-gram as a continuous vector and learn these representations in the context of the authorship attribution tasks [43].

Hitchler et al. propose a CNN based on pre-trained embedding word vector concatenated with one-hot encoding of POS tags; however, they have not shown any ablation study to report the contribution of POS tags on the final performance results [19]. Zhang et al. introduce a syntax encoding approach using convolutional neural networks which combines with a lexical models, and applies it to the domain of authorship attribution [57]. We propose a simpler yet more effective way of encoding syntactic information of documents for the domain of authorship attribution [22]. Moreover, we employ a hierarchical neural network to capture the structural information of documents and finally introduce a neural model which incorporates all three stylistic features including lexical, syntactic, and structural [21].

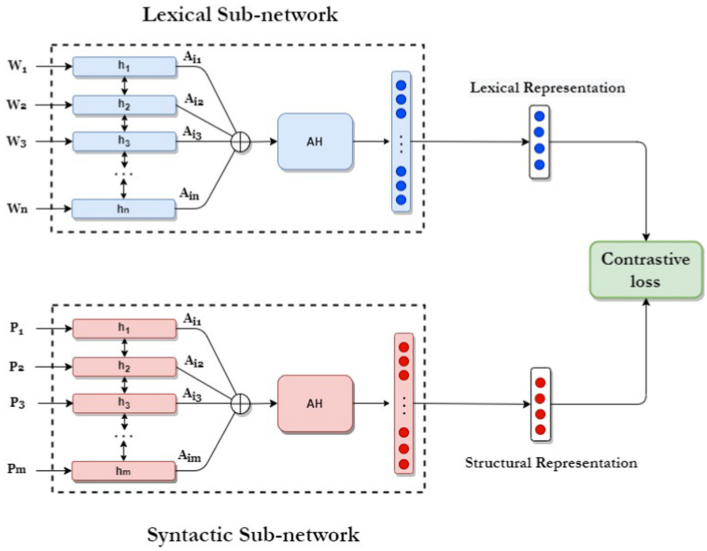

Fig. 1. The overall architecture of LexicoSynt network.

3 PROPOSED FRAMEWORK: LEXICOSYNT NETWORK

The LexicoSynt network is a Siamese network [10] which comprises lexical and syntactic sub-networks and a loss function. Figure 1 illustrates the overall architecture of framework. The input to the system is a pair of sequences (lexical and syntactic) and a label. The lexical and syntactic sequences are passed through the lexical and syntactic sub-networks, respectively, yielding two outputs which are passed to the cost module. In what follows, we elaborate each component.

3.1 Lexical Sub-Network

The lexical sub-network encodes the sequence of words in a sentence (Figure 2(a)). Each word in the sentence (denoted as S) is embedded into a trainable vector representation (W_i) and is fed into the lexical sub-network. This network consists of two main parts; bidirectional LSTM (H which is concatenation of forward LSTM $\overrightarrow{h_t}$ and backward LSTM $\overleftarrow{h_t}$) and a self-attention mechanism (A) proposed by Lin et al. [31]. The self-attention mechanism provides a set of summation weight vectors (W_{s2},W_{s1}) which are dotted with the LSTM hidden states, resulting weighted hidden states (M). Finally, a pooling layer followed by a multilayer perceptron is used to generate the final vector representation.

$$S = (W_1, W_2, \ldots, W_n),$$
$$\overrightarrow{h_t} = \overrightarrow{LSTM}(W_t, \overrightarrow{h_{t-1}}),$$
$$\overleftarrow{h_t} = \overleftarrow{LSTM}(W_t, \overleftarrow{h_{t+1}}),$$
$$H = (h_1, h_2, \ldots, h_n),$$
$$A = softmax(W_{s2} tanh(W_{s1} H^T)),$$
$$M = AH.$$

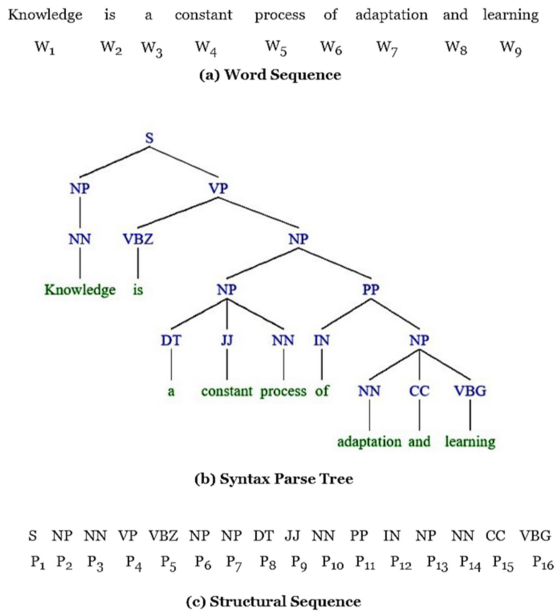

Fig. 2. An example of an input sentence and the corresponding (a) word sequence representation, (b) syntax parse tree representation, and (c) linearized parse tree representation known as structural sequence.

3.2 Syntactic Sub-Network

The syntactic sub-network aims at encoding the syntactic information of sentences. For this purpose, we use syntax parse trees. Syntax parse trees represent the syntactic structure of a given sentence. An example of such a syntax tree is given in Figure 2(b). To adapt the tree representation to recurrent neural networks, we linearize the syntax parse tree to a sequence of structural labels. Figure 2(c) shows the structural label sequence of Figure 2(b) following a depth-first traversal order. Each structural label in the sequence is embedded into a trainable vector representation and subsequently is fed into the syntactic sub-network which has an identical architecture as the lexical sub-network. Needless to say the structural sequence which is the linearized syntax parse tree is longer than the word sequence of a sentence.

3.3 Loss Function

Given a sentence, we aim at minimizing the distance of two learned vector representations from the lexical and syntactic sub-networks. The distance of two vectors (d_n) are maximized if they are not representations of an identical sentence. In other words, the output of the lexical and the syntactic sub-networks are similar ($y_n = 1$) for genuine pairs (positive samples), and different ($y_n = 0$) for false pairs (negative samples). We propose to use the contrastive loss, originally proposed for training of Siamese networks [10], as the following:

$$E = \frac{1}{2N} \sum_{n=1}^{N} y_n d_n^2 + (1 - y_n) max(margin - d_n, 0)^2,$$

where:

$$d_n = ||V_{lexical} - V_{structural}||_2.$$

$V_{lexical}$ and $V_{structural}$ are the learned sentence representations from lexical and syntactic sub-networks, respectively, and d_n is the Euclidean distance of $V_{lexical}$ and $V_{structural}$. $y \in [0, 1]$ is the binary similarity metric between the input pairs; y is 1 if the syntactic and lexical pairs belong to an identical sentence and 0 if they belong to different sentences. Parameter margin is the minimum distance between negative pairs and is set to 1 in our implementation. Hence, the negative samples only contribute to the loss if their distance is less than the margin.

Training the network by such objective function generates similar vector representations for sentences with identical syntactic structures but different semantics. In other words, the contrastive loss objective pushes the sentences with identical syntactic structures close to each other in the embedding space. As a result, the learned vector representations from the lexical network does not mainly carry semantic relationships any more, but more structural information. Finally, the learned structural representations of words from this self-supervised framework can be simply used as complementary information to the existing ordinary pre-trained word embeddings to better suit the neural models for the domain of authorship attribution.

4 EXPERIMENTAL STUDIES

In order to evaluate the effectiveness of our proposed framework, we conduct two sets of evaluations: intrinsic evaluation and extrinsic evaluation. In the former, we investigate the different linguistic properties captured by the learned structural representations and compare them against the existing pre-trained word embeddings. In the latter, we utilize the learned structural representations for the domain of authorship attribution and compare it against the existing baselines. In what follows, we elaborate the implementation details and the experimental configurations. To make the experiments reported in this article reproducible, we have made our framework implementations publicly available.[1]

4.1 Data

4.1.1 Training Data. The proposed model has been trained on **LAMBADA** (**LAnguage Modeling Broadened to Account for Discourse Aspects**) dataset [37] which contains the full text of 2,662 unpublished novels from 16 different genres. The fact that the training data comes from the wide range of genres maximizes the potential efficacy for learning diverse sentence structures. The total number of sentences in the training set is 14,746,838 where 1,000 sentences are randomly selected for the development set.

4.1.2 Test Data. We evaluate the proposed approach on the following authorship attribution benchmark datasets:

- **CCAT10, CCAT50:** Newswire stories from **Reuters Corpus Volume 1** (**RCV1**) written by 10 and 50 authors, respectively [50].
- **BLOGS10, BLOGS50:** Posts written by 10 and 50 top bloggers, respectively, originated from dataset of 681,288 blog posts by 19,320 bloggers for blogger.com [44].

4.2 Training

For the input data of the syntactic sub-network, we have generated the parse tree of each sentence in the training set using CoreNLP parser [33]. Each sentence and its corresponding linearized parse tree is fed into the network as a genuine (or positive) pair. To generate the false (negative) pairs, we have paired each sentence in the batch with a randomly selected linearized parse tree in

[1] https://github.com/fereshtehjafarii/StructuralSentenceRepresentation.

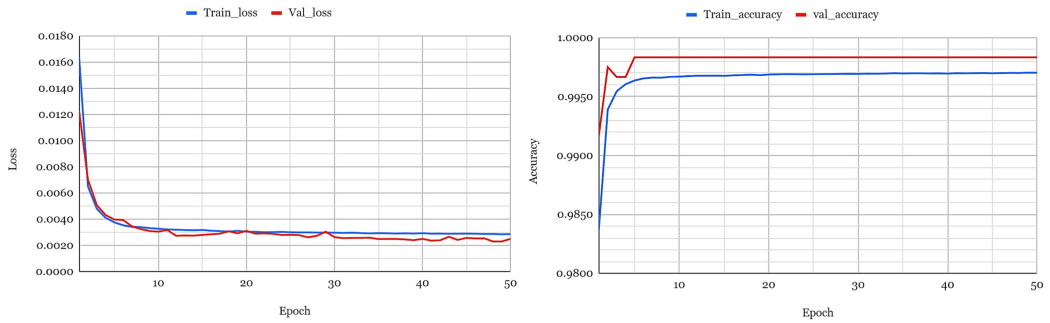

Fig. 3. The training and validation loss and accuracy over 50 epochs of training.

the same batch. Hence, the number of training samples is twice as the number of sentences in the training set ($14{,}746{,}838 \times 2 = 29{,}493{,}676$). For the validation set, 1,000 pairs are chosen randomly.

We have used batch size of 400 and learning rate of 5e-4 with Adam optimizer for all the experiments. All the weights in the neural networks are initialized using uniform Xavier initialization. The both lexical and structural embeddings are initialized from $U[-0.1, 0.1]$ and their dimension has been set to 300 and 100, respectively. We limit the maximum input length for lexical and syntactic sub-networks to 40 and 80, respectively. The inventory of structural labels include 77 phrase labels. Figure 3 illustrates the loss and accuracy of the model across 50 epochs of training, respectively. As shown in the figures, the training loss converges to 0.0029, the training accuracy is 99.70%, and the validation loss and accuracy are 0.0023 and 99.83%, respectively. The low value of loss indicates that positive samples are successfully encoded to similar representations and negative samples are encoded to different representations. The model performs better on the validation set compared to the training set primarily due to the fact that the training set is significantly larger ($\sim$29,000 times) than the validation set. This is inevitable in our training scenario since training contrastive loss objective requires a huge amount of data in order to learn the proper representations.

4.3 Representation Learning Evaluation: Probing Tasks

We use 10 probing tasks introduced by Conneau et al. [12] to investigate the linguistic features that the learned structural representations capture. We use the structural embeddings of words to create the **Bag of Vectors** (**BoV**) representation for each sentence and we evaluate these sentence embeddings in each task. The experiments are configured according to the recommended settings using logistic regression.[2] The probing tasks are grouped into three classes including surface information, syntactic information, and semantic information. The tasks in the surface information group do not require any linguistic knowledge while the tasks in the syntactic information group test if the sentence embeddings are sensitive to the syntactic properties. The tasks in the semantic information group not only rely on syntactic information but also require understanding of semantics about sentences. In what follows, we elaborate each probing task and report the corresponding evaluation results.

4.3.1 *Surface Information.*

— *SentLen*: to predict the length of sentences in terms of number of word.
— *WC*: to recover the information about the original word from its embedding.

[2] https://github.com/facebookresearch/SentEval.

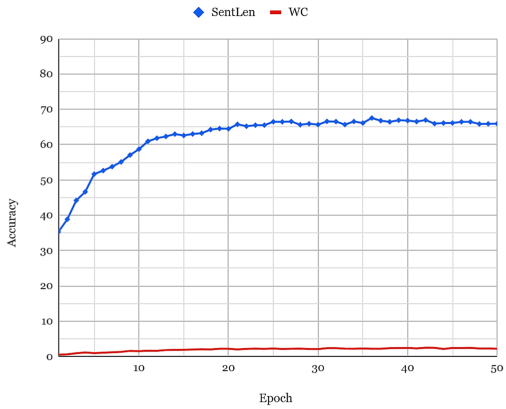

Fig. 4. The accuracy of learned surface information over 50 epochs of training.

Figure 4 illustrates how surface information accuracy changes in the function of training epochs. According to the figure, the sentence length accuracy increases with epochs. Unsurprisingly, the WC accuracy is mostly flat and about 2.42% since the model encodes structural information rather than semantic information. In other words, in this model, all the words with an identical syntactic role in the sentence (for instance Nouns) are mapped to a single identical vector. This way of encoding is an n-to-1 mapping; hence, recovering the original word from its structural embedding is almost impossible. However, concatenating the structural embedding of words with their general embeddings (BoV structural+FastText and BoV structural+Glove) enhances the performance of WC compared to when only their general embeddings are used (Table 1). This implies that structural information of words in the sentence improves the recovery of its content.

4.3.2 *Syntactic Information.*

— *BShift*: to predict if the two adjacent words in the sentences were inverted. This task tests if the encoder is sensitive to the word order.
— *TreeDepth*: to classify sentences based on the depth of the longest path from the root to any leaf. This task investigates if the encoder infers the hierarchical structure of sentences.
— *TopConst*: to classify the sentences based on the sequence of top constituencies immediately below the sentence. This task tests the ability of encoder in capturing the latent syntactic structure.

Figure 5 illustrates how syntactic information accuracy changes in terms of training epochs. According to the figure, TreeDepth and TopConst performance keep increasing with epochs. However, BShift curve is mostly flat, suggesting that what Bidirectional LSTM is able to capture about this task is already encoded in its architecture and further training does not help much.

4.3.3 *Semantic Information.*

— *Tense*: to predict the tense of the main-clause verb. This task tests if the encoder captures the structural information about the main clause.
— *SubjNum*: to predict the number of the subject of main clause. This task tests if the encoder captures the structural information about the main clause and its arguments.
— *ObjNum*: to predict the number of direct object of the main clause. This task tests if the encoder captures the structural information about the main clause and its arguments.

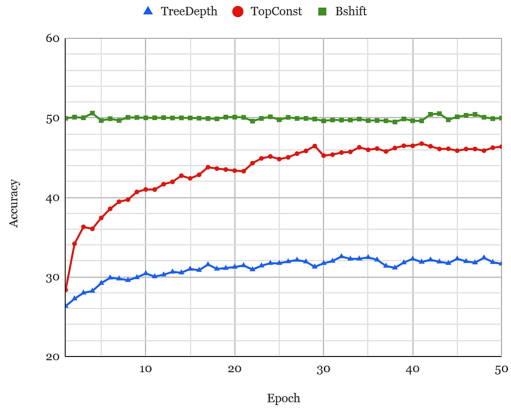

Fig. 5. The accuracy of syntactic information over 50 epochs of training.

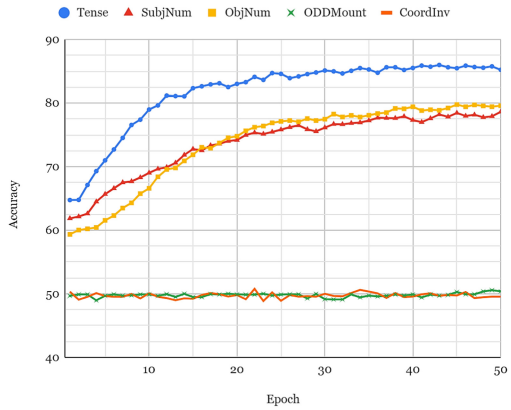

Fig. 6. The accuracy of semantic information over 50 epochs of training.

— *SOMO*: to predict whether an arbitrary chosen noun or verb in the sentences has been modified or not. This task tests if the encoder has captured semantic information to some extent.
— *CoordInv*: to predict whether the order of clauses in the sentences are intact or modified. This task tests if the encoder has the understanding of broad discourse and pragmatic factors.

Figure 6 illustrates how the accuracy of semantic information captured by the model changes in function of training epochs. According to the figure, the accuracy of Tense, SubjNum, and ObjNum increases when the number of epochs increase. It is worth mentioning that the accuracy of these probing tasks, which heavily rely on structural information of sentences, show more increase during the training process. On the other hand, SOMO and CoordInV which mostly rely on semantic information have flat curves, indicating that the further training does not improve their performance. This is clearly due to the fact that these two tasks deeply rely on semantic information of sentences while structural embeddings lack such information.

An interesting observation is that the structural representations generally demonstrate a better performance in the tasks where both semantic and structural information is required (e.g., Tense,

Table 1. Probing Task Accuracies for Different Sentence Representations

Model	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Baseline Representations										
Majority vote	20.0	0.5	17.9	5.0	50.0	50.0	50.0	50.0	50.0	50.0
Hum. Eval.	100	100	84.0	84.0	98.0	85.0	88.0	86.5	81.2	85.0
BoV Glove	58.1	75.5	30.0	49.7	49.8	83.8	77.2	76.3	49.4	49.9
BoV FastText	53.3	79.8	31.0	52.6	50.1	86.7	79.2	79.4	50.2	50.0
Our Proposed Structural Representations										
BiLSTM Structural	**77.8**	0.2	23.7	17.2	**50.3**	57.2	50.3	51.8	49.9	**52.4**
BoV Structural	66.1	2.4	32.3	45.9	50.1	85.5	78.5	79.8	50.3	49.8
BoV Structural+Glove	64.0	75.1	31.3	54.5	50.0	85.4	80.7	81.9	50.1	50.4
BoV Structural+FastText	64.0	**79.9**	**33.6**	**57.1**	50.1	**87.3**	**81.2**	**82.1**	**50.8**	50.1

SubjNum, and ObjNum) compared to the tasks that either only rely on syntactic information (e.g., TreeDepth, TopConst) or semantic information (e.g., SOMO, CoordInv). This feature can be due to the co-supervision of lexical and syntactic sub-networks in the representation learning process.

4.4 Representation Learning Evaluation: Comparing to the Baselines

In this section, we compare the performance of structural embeddings learned from our model to two other pre-trained general word embeddings, including Glove [39] and FastText [7]. We use BoV representation of sentences for both its simplicity and its capability at capturing sentence level properties [12]. Table 1 reports the results of different sentence embeddings for all the 10 probing tasks and the best results are highlighted in bold. Human. Eval. results report the human validated upper bounds for all the tasks (refer to [12] for more details). In BiLSTM Structural, we have used the syntactic sub-network as the encoder. Using BiLSTM encoder to generate the structural sentence embeddings does not show any improvement in terms of accuracy when compared to BoV structural representations except for BShift (0.07% increase) and CoordInv (2.65% increase) simply due to the fact that these tasks are heavily sensitive to the word orders in the sentence and BiLSTM preserves orders in the input sequence while BoV does not.

The performance results of word embeddings alone show that FastText outperforms Glove in all the tasks by the average of 1.26%. This indicates that FastText embeddings capture slightly more linguistic features compared to Glove. Moreover, concatenating Glove/FastText embeddings with the structural embeddings (BoV Structural+Glove/BoV Structural+FastText) improves the the performance in all the tasks by the average of 2.37%/2.39%. Hence, concatenating structural embeddings with the pre-trained word embeddings further improves the linguistic features captured compared to when the word embeddings are used alone.

According to the results, BoV representation of sentences from structural embeddings outperforms FastText embeddings in SentLen and TreeDepth by 12.8% and 1.3%, respectively. Furthermore, combining structural embeddings and FastText embeddings enhances the accuracy in all tasks compared to when only either of them is used. For instance, it improves the accuracy of ObjNum by 2.7%, SubjNum by 2.00%, TopConst by 4.5%, TreeDepth by 2.6%, SentLen by 10.7% compared to when only FastText embeddings are used. Unsurprisingly, combining FastText embeddings with structural embeddings does not significantly improve accuracy in WC, SOMO, and CoordInv tasks due to the fact that these tasks are heavily reliant on semantics and structural embeddings do not result in further improvements. Finally, BoV representation of sentences by combining structural embedding and FastText embeddings consistently outperforms the baseline representations in all the tasks, with an average improvement over BoV Glove and BoV FastText of 6% and 3.9%, respectively.

Table 2. Probing Task Accuracies for Different Model Configurations

Model config	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Model Architecture for Structural Representation Learning										
NoATT_seq4040	61.6	5.3	29.7	43.0	49.7	85.2	69.5	71.7	49.9	49.7
WeightedATT_seq4040	62.6	5.1	30.4	43.4	50.0	85.3	68.8	71.4	49.9	49.0
SelfATT_AVGPool_seq4040	62.8	**6.6**	31.4	45.7	49.7	**86.2**	71.9	74.4	49.9	49.7
SelfATT_MaxPool_seq4040	65.9	2.1	31.6	45.4	49.9	85.2	77.9	79.5	50.0	49.5
SelfATT_MaxPool_seq4080	**66.1**	2.4	**32.3**	**45.9**	**50.1**	85.5	**78.5**	**79.8**	**50.3**	**49.8**

4.5 Model Selection

We have performed an ablation study on different components of the model: the self-attention mechanism, the pooling mechanism, and the length of structural sequences. Table 2 reports the result of our experiments. In the NoATT_seq4040 configuration, we do not use any attention mechanism and set the length of both lexical and structural sequence to 40. In WeightedATT_seq4040, we use the traditional attention mechanism [4]. In SelfATT_AVGPool_seq4040 and SelfATT_MaxPool_seq4040, we incorporate Self-attention mechanism [31] and use average-pooling and max-pooling respectively to generate the final representations. Finally, in SelfATT_MaxPool_seq4080, we use self-attention mechanism with max-pooling where the length of lexical and structural sequence is 40 and 80, respectively . In all of the configurations other components of the network including BiLSTM and loss function, have been kept identical. According to the results, using self-attention mechanism improves the performance in most of the tasks compared to when no attention or traditional attention mechanism is used. When using self-attention mechanism, max-pooling performs better than average-pooling in most of the tasks. We observe that increasing the length of the structural sequence to 80 slightly improves the performance. This is due to the fact that structural sequence, which is a linearized syntax parse tree, is longer than the original sentence. Ultimately, the self-attention mechanism with max-pooling, and the structural sequence of length 80 is used as the final configuration (SelfATT_MaxPool_seq4080).

4.6 Test on Authorship Attribution Datasets

In our previous work [21], we have introduced a neural network which encodes the stylistic information of documents from three levels of language production (lexical, syntactic, and structural). First, we obtain both lexical and syntactic representation of words using lexical and syntactic embeddings as shown in Figure 7. For lexical representations, we embed each word into a pre-trained Glove embeddings and represent each sentence as the sequence of its corresponding word embeddings. For syntactic representations, we convert each word into the corresponding POS tag in the sentence, and then embed each POS tag into a low dimensional vector $P_i \in \mathbb{R}^{d_p}$ using a trainable lookup table $\theta_P \in \mathbb{R}^{|T| \times d_p}$, where T is the set of all possible POS tags in the language. Subsequently, these two representations are fed into two identical hierarchical neural networks which encode the lexical and syntactic patterns of documents independently and in parallel. Ultimately, these two representations are aggregated into the final vector representation of the document which is fed into a softmax layer to compute the probability distribution over class labels.

We have compared our proposed style-aware neural model (Style-HAN) with the other stylometric models in the literature, including Continuous N-gram representation [43], N-gram CNN [47], and syntax-CNN [57]. Table 3 reports the accuracy of the models on the four benchmark datasets. All the results are obtained from the corresponding papers, with the dataset configuration kept identical for the sake of fair comparison. In Syntactic-HAN [21], only syntactic representation of documents is fed into the softmax layer to compute the final predictions. Similarly, in Lexical-HAN [21], only lexical representation of documents is fed into the softmax classifier. The final stylometry

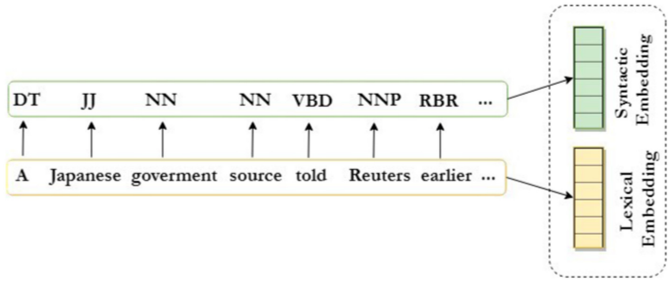

Fig. 7. Lexical and syntactic embedding.

Table 3. The Accuracy of Different Models for All Datasets

Model	CCAT10	CCAT50	BLOGS10	BLOGS50
Baselines				
Continuous n-gram	74.8	72.6	61.3	52.8
N-gram CNN	86.8	76.5	63.7	53.0
Syntax-CNN	88.2	81.0	64.1	56.7
Lexical-HAN	86.0	79.5	70.8	59.8
Our Proposed Models				
Syntactic-HAN	63.1	41.3	69.3	57.8
Syntactic+Lexical-HAN (Style-HAN)	90.6	82.3	72.8	61.2
Structural-HAN	65.4	45.2	70.6	59.5
Structural+Lexical-HAN	**92.4**	**83.2**	**73.5**	**61.7**

model, Style-HAN, fuses both representations. In order to examine the efficacy of our proposed structural embeddings in this article against the previously proposed POS-encoding (Syntactic-HAN) and style-aware neural network (Style-HAN), we adopt the same neural network architecture with two different settings; (1) using only structural embedding of the words (Structural-HAN) and (2) using pre-trained Glove word embeddings concatenated with the structural embeddings (Structural+Lexical-HAN). We chose to use Glove (instead of FastText) for our performance study in order to have a fair comparison with the current state-of-the-art Lexical-HAN and Style-HAN methods, which used Glove embeddings as their lexical embeddings.

In Table 3, the best performance result for each dataset is highlighted in bold. It shows that the proposed Structural+Lexical-HAN consistently outperforms all the baselines. The average improvement over Continuous n-gram, N-gram CNN, Syntax-CNN, and Lexical-HAN is 18.9%, 11%, 7.2%, and 5%, respectively. These significant results confirm that explicit representation learning of syntactic structure of sentences improves the performance of lexical-based neural models in the task of authorship attribution. Among the proposed models, Structural-HAN constantly outperforms Syntactic-HAN. This observation indicates the effectiveness of the learned structural representation of words in our proposed self-supervised framework. It is worth mentioning that the learned structural embeddings from our self-supervised framework not only improves the performance of the syntactic neural model but also eliminates the necessity of syntactic parsing in the sentence representation step in style-aware neural network; hence, it is computationally more efficient. Finally, Structural+Lexical-HAN is consistently the best among the proposed models across all datasets.

It is worth mentioning that in Syntactic-HAN, the syntactic units are POS tags which are embedded into randomly initialized vector representations. These syntactic representations are learned during the training phase of the Syntactic-HAN model on the authorship attribution datasets. However, in the LexicoSynt network, the units in the syntactic sub-network are POS tags which are sequenced based on the syntactic structure of sentences (linearized syntax parse tree). These structural units have been pre-trained on almost 29 million sentences in the LAMBADA dataset and subsequently used as the initialization for the Structural-HAN model. Hence, structural representations are pre-trained vector representations of POS tags which carry both structural and syntactic features of sentences. Combining structural and syntactic features of sentences is one of the advantages of Structural-HAN.

The explicit representation learning of sentence structure using the LexicoSynt network has improved the performance results of the previously proposed model in the authorship attribution task; however, learning the structural representations can be improved in different ways. The current design of the LexicoSynt network utilizes a fixed length for the lexical and structural sequences. Truncating and padding the sentences to fit this fixed length can affect their intended semantics and structures. An adaptive approach that can tailor this length to the specific case is desirable. Another limitation of the current design is due to its training on the dataset of a variety of novels. Even though the training dataset for the LexicoSynt network contains numerous novels from 16 different genres and ensures the learning of diverse sentence structures, this might limit its applicability to other domains. Domain-specific representation learning can address this limitation.

5 CONCLUSION

In this article, we have proposed a self-supervised framework for learning structural representation of sentences for the domain of authorship attribution. The result of training this self-supervised framework is pre-trained structural embeddings which capture information regarding the syntactic structure of sentences. Subsequently, these structural embeddings can be concatenated to the existing pre-trained word embeddings and create a style-aware embedding which carries both semantic and syntactic information and it is well-suited for the domain of authorship attribution. Moreover, structural embeddings eliminate the necessity of syntactic parsing for training syntactic neural networks; therefore, training a neural model using pre-trained structural embeddings is computationally more efficient. According to our experimental results on four benchmark datasets in authorship attribution, using structural embedding improves the performances of the proposed neural model.

REFERENCES

[1] Sadia Afroz, Michael Brennan, and Rachel Greenstadt. 2012. Detecting hoaxes, frauds, and deception in writing style online. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. IEEE, 461–475.

[2] Shlomo Argamon-Engelson, Moshe Koppel, and Galit Avneri. 1998. Style-based text categorization: What newspaper am I reading. In *Proceedings of the AAAI Workshop on Text Categorization*. 1–4.

[3] Douglas Bagnall. 2016. Authorship clustering using multi-headed recurrent neural networks. arXiv:1608.04485 . Retrieved from https://arxiv.org/abs/1608.04485

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473. Retrieved from https://arxiv.org/abs/1409.0473

[5] Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6008–6019.

[6] Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 14–19.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.
[8] Michael Brennan, Sadia Afroz, and Rachel Greenstadt. 2012. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security* 15, 3 (2012), 12.
[9] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. arXiv:1803.11175 . Retrieved from https://arxiv.org/abs/1803.11175
[10] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1, IEEE, 539–546.
[11] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 670–680.
[12] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single \&# vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Vol. 1, Association for Computational Linguistics, 2126–2136.
[13] Walter Daelemans. 2013. Explanation in computational stylometry. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 451–462.
[14] Zhenhao Ge, Yufang Sun, and Mark J. T. Smith. 2016. Authorship attribution using a neural network language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 4212–4213.
[15] Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative learning of disjoint syntax and semantics. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 1118–1128.
[16] Maryam Heidari and James H. Jones. 2020. Using bert to extract topic-independent sentiment features for social media bot detection. In *Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*. IEEE, 0542–0547.
[17] M. Heidari, J. H. Jones, and O. Uzuner. 2020. Deep contextualized word embedding for text-based online user profiling to detect social bots on Twitter. In *Proceedings of the 2020 International Conference on Data Mining Workshops*. 480–487. DOI: https://doi.org/10.1109/ICDMW51313.2020.00071
[18] John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4129–4138.
[19] Julian Hitschler, Esther van den Berg, and Ines Rehbein. 2017. Authorship attribution with convolutional neural networks and POS-eliding. In *Proceedings of the Workshop on Stylistic Variation*. 53–58.
[20] Fereshteh Jafariakinabad and Kien A. Hua. 2016. Maximal sequence mining approach for topic detection from microblog streams. In *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence*. IEEE, 1–8.
[21] Fereshteh Jafariakinabad and Kien A. Hua. 2019. Style-aware neural model with application in authorship attribution. In *Proceedings of the 2019 18th IEEE International Conference on Machine Learning and Applications*. IEEE, 325–328.
[22] Fereshteh Jafariakinabad, Sansiri Tarnpradab, and Kien A. Hua. 2020. Syntactic neural model for authorship attribution. In *Proceedings of the 33rd International Flairs Conference*.
[23] Jad Kabbara and Jackie Chi Kit Cheung. 2016. Stylistic transfer in natural language generation systems using recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*. 43–47.
[24] Vishal Kaushal and Manasi Patwardhan. 2018. Emerging trends in personality identification using online social networks-a literature survey. *ACM Transactions on Knowledge Discovery from Data* 12, 2, Article 15 (Jan. 2018), 30 pages. DOI: https://doi.org/10.1145/3070645
[25] Moniba Keymanesh, Saket Gurukar, Bethany Boettner, Christopher Browning, Catherine Calder, and Srinivasan Parthasarathy. 2020. Twitter watch: Leveraging social media to monitor and predict collective-efficacy of neighborhoods. In *Complex Networks XI*. H. Barbosa, J. Gomez-Gardenes, B. Gonçalves, G. Mangioni, R. Menezes, and M. Oliveira (Eds.). Springer, 197–211.
[26] Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. 3294–3302.
[27] Markus Krause. 2014. A behavioral biometrics based authentication method for MOOC's that is robust against imitation attempts. In *Proceedings of the 1st ACM Conference on Learning@ Scale Conference*. ACM, 201–202.
[28] Tim Kreutz and Walter Daelemans. 2018. Exploring classifier combinations for language variety identification. In *Proceedings of the 5th Workshop on NLP for Similar Languages, Varieties and Dialects*. 191–198.

[29] Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1426–1436.

[30] Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 688–697.

[31] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. arXiv:1703.03130. Retrieved from https://arxiv.org/abs/1703.03130.

[32] Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. 2017. Structural embedding of syntactic trees for machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 815–824.

[33] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60.

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. 3111–3119.

[35] Matthew L. Newman, James W. Pennebaker, Diane S. Berry, and Jane M. Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and Social Psychology Bulletin* 29, 5 (2003), 665–675.

[36] Toktam A. Oghaz, Ece Çiğdem Mutlu, Jasser Jasser, Niloofar Yousefi, and Ivan Garibay. 2020. Probabilistic model of narratives over topical trends in social media: A discrete time model. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*. ACM, 281–290. DOI: https://doi.org/10.1145/3372923.3404790

[37] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1525–1534.

[38] James W. Pennebaker and Laura A. King. 1999. Linguistic styles: Language use as an individual difference. *Journal of Personality and Social Psychology* 77, 6 (1999), 1296.

[39] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[40] Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. arXiv:1806.06259. Retrieved from https://arxiv.org/abs/1806.06259.

[41] Juan-Pablo Posadas-Durán, Ilia Markov, Helena Gómez-Adorno, Grigori Sidorov, Ildar Batyrshin, Alexander Gelbukh, and Obdulia Pichardo-Lagunas. 2015. Syntactic n-grams as features for the author profiling task. In *Proceedings of the Working Notes Papers of the CLEF*.

[42] Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. 2010. Authorship attribution using probabilistic context-free grammars. In *Proceedings of the ACL 2010 Conference Short Papers*. Association for Computational Linguistics, 38–42.

[43] 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*. 267–273.

[44] Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2006. Effects of age and gender on blogging. In *Proceedings of the AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*. Vol. 6, 199–205.

[45] Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. arXiv:1702.01841. Retrieved from https://arxiv.org/abs/1702.01841.

[46] Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. arXiv:1711.02013. Retrieved from https://arxiv.org/abs/1711.02013.

[47] Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, 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*. 669–674.

[48] Juan Soler and Leo Wanner. 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*. Vol. 2, 681–687.

[49] Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*. 1717–1729.

[50] Efstathios Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management* 44, 2 (2008), 790–799.
[51] Kalaivani Sundararajan and Damon Woodard. 2018. What represents "style" in authorship attribution? In *Proceedings of the 27th International Conference on Computational Linguistics*. 2814–2822.
[52] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. arXiv:1905.06316. Retrieved from https://arxiv.org/abs/1905.06316.
[53] Chris van der Lee and Antal van den Bosch. 2017. Exploring lexical and syntactic features for language variety identification. In *Proceedings of the 4th Workshop on NLP for Similar Languages, Varieties and Dialects*. 190–199.
[54] William Yang Wang. 2017. "Liar, Liar Pants on Fire": A new benchmark dataset for fake news detection. arXiv:1705.00648. Retrieved from https://arxiv.org/abs/1705.00648.
[55] Reza Zafarani, Lei Tang, and Huan Liu. 2015. User identification across social media. *ACM Transactions on Knowledge Discovery from Data* 10, 2, Article 16 (Oct. 2015), 30 pages. DOI: https://doi.org/10.1145/2747880
[56] Meishan Zhang, Zhenghua Li, Guohong Fu, and Min Zhang. 2019. Syntax-enhanced neural machine translation with syntax-aware word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1151–1161.
[57] 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*. 2742–2753.

Received October 2020; revised June 2021; accepted October 2021