Machine Learning Techniques for Topic Detection and Authorship Attribution in Textual Data

Fereshteh Jafariakinabad
University of Central Florida

MACHINE LEARNING TECHNIQUES FOR TOPIC DETECTION AND AUTHORSHIP ATTRIBUTION IN TEXTUAL DATA

by

FERESHTEH JAFARIAKINABAD
B.S. Amirkabir University of Technology, 2013

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2021

Major Professor: Kien A. Hua

ABSTRACT

The unprecedented expansion of user-generated content in recent years demands more attempts of information filtering in order to extract high-quality information from the huge amount of available data. In this dissertation, we begin with a focus on topic detection from microblog streams, which is the first step toward monitoring and summarizing social data. Then we shift our focus to the authorship attribution task, which is a sub-area of computational stylometry. It is worth mentioning that determining the style of a document is orthogonal to determining its topic, since the document features which capture the style are mainly independent of its topic.

We initially present a frequent pattern mining approach for topic detection from microblog streams. This approach uses a Maximal Sequence Mining (MSM) algorithm to extract pattern sequences, where each pattern sequence is an ordered set of terms. Then we construct a pattern graph, which is a directed graph representation of the mined sequences, and apply a community detection algorithm to group the mined patterns into different topic clusters. Experiments on Twitter datasets demonstrate that the MSM approach achieves high performance in comparison with the state-of-the-art methods.

For authorship attribution, while previously proposed neural models in the literature mainly focus on lexical-based neural models and lack the multi-level modeling of writing style, we present a syntactic recurrent neural network to encode the syntactic patterns of a document in a hierarchical structure. The proposed model learns the syntactic representation of sentences from the sequence of part-of-speech tags. Furthermore, we present a style-aware neural model to encode document information from three stylistic levels (lexical, syntactic, and structural) and evaluate it in the domain of authorship attribution. Our experimental results, based on four authorship attribution benchmark datasets, reveal the benefits of encoding document information from all three stylistic levels when compared to the baseline methods in the literature. We extend this work and adopt a transfer learning approach to measure the impact of lower-level linguistic representations versus higher-level linguistic representations on the task of authorship attribution.

Finally, we present a self-supervised framework for learning structural representations of sentences. The self-supervised network is a Siamese network with 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. This model is trained based on a contrastive loss objective. As a result, each word in the sentence is embedded into a vector representation which mainly carries structural information. The learned structural representations can be concatenated to the existing pre-trained word embeddings and create style-aware embeddings that carry both semantic and syntactic information and is well-suited for the domain of authorship attribution.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Kien A. Hua for his guidance and support throughout the process of completing my PhD research. I would also like to thank my dissertation committee members, Dr. Yanjie Fu, Dr. Liqiang Wang, Dr. Beth Young, for their efforts in serving in my dissertation committee and providing valuable guidance and suggestions on my dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Topic Detection

The term *"microblogging"* was coined in 2006-2007 and since then it has been used to describe social media where users are able to share small units of content. The popularity of online microblogging social media in recent years has led to unprecedented growth in user-generated content, which is a rich source of information about real-world events. The availability of this huge amount of data has initiated research on extracting high quality information by monitoring and analyzing microblogging streams. The very first step towards extracting and summarizing useful information from social streams is Topic Detection. Early work on Topic Detection and Tracking (TDT), which was introduced in the late nineties, studied events in news streams [4]. TDT deals with detection and tracking of events from the stream of stories. The input stream may or may not be pre-segmented into stories and the events may or may not be known to the system; in other words, the system may or may not be trained to recognize a specific event [4]. Detecting unknown events from streams of stories is more challenging due to lack of any prior knowledge about the event. Even though numerous methods of event detection for conventional news media have been proposed in TDT, the noise of user-based contents and their short length, as well as heterogeneous characteristics of social data streams make it a more challenging task when compared to news streams.

General textual topic detection methods are classified into three classes: document-pivot methods, feature-pivot methods, and probabilistic topic models. In this work, we focus on feature-pivot methods, which cluster terms with respect to their co-occurrence in the corpus. Most of the methods that fall under this category leverage pairwise co-occurrences of terms which yield merged topics in a corpus containing interconnected topics. Frequent Pattern Mining (FPM) is one of the approaches which aims to address this issue by examining simultaneous co-occurrence of more than two terms [33]. Soft Frequent Pattern Mining (SFPM) is a modified version of FPM where a large number of terms must co-occur frequently, but not necessarily all, leading to a soft version of FPM [66]. In this study, we aim to incorporate relative positional information of terms, as well as distances between terms in a sequence. We argue that this strategy reduces the likelihood of extracting incorrect correlations of terms because the pattern mining is based on term sequence as a pattern which carries more semantic information about the content than an unordered list of terms can. This improvement leads to more accurate topic detection results.

In general, any topic detection method which is based on statistical inferences is heavily reliant on long documents, while user generated content in social media is usually in the form of short texts. Aggregation is a common solution for addressing this problem in information retrieval. Luca et al. [2] explored the effect of preprocessing steps and the topic detection algorithm itself on social streams. According to their experiments, in most cases the time-aggregated datasets achieve lower topic recall scores than non-aggregated datasets. The underlying reason behind this is that the aggregated tweets may represent a mixture of topics rather than a single topic and are therefore more likely to indicate an incorrect association of words. Transaction-level pattern mining not only results in more informative patterns, but also decreases the likelihood of generating mixed topics since it examines the co-occurrence of terms in the transaction rather than document. Hence, aggregation of tweets does not affect the algorithm in this case. In order to reduce information redundancy in mined patterns we utilize a maximal patterns scheme.

Authorship Attribution

Individuals express their thoughts in different ways due to many factors including the conventions of the language, educational background, and the intended audience, etc. In written language, the combination of consistent conscious or unconscious decisions in language production, known as writing style, has been studied widely [47, 57]. Computational stylometry consists of three tasks; *authorship attribution* (to determine the likelihood of a particular author having written a given piece of text), *authorship profiling* (to detect characteristics of the author that has produced a given piece of text e.g. gender, age, personality,...) and *similarity detection* (to compare multiple pieces of work and decide if they have written by a single author or not). Stylistic features are generally *content-independent*. They are consistent across different documents written by a specific author (or author groups). Lexical, syntactic, and structural features are three main families of stylistic features. Lexical features represent author's character and word use preferences, while syntactic features capture the syntactic patterns of the sentences in a document. Structural features reveal information about how an author organizes the sentences in a document.

To date, the existing approaches in the domain of authorship attribution fall into two categories. The first category adopts traditional machine learning techniques to identify the author of a given document. In this approach the stylistic features are engineered and extracted from the documents and are subsequently used as the inputs of different kind of classifiers [81, 20, 89, 95, 69, 82]. These features reveal statistical information of documents in lexical, syntactic, and structural levels. For instance, frequency of certain words, character distribution, function word distribution, frequency of part of speech tags, the number of sentences per paragraph, etc. A limitation of this approach is that the feature extracting process ignores rich sequential information in the sentences and the document.

The second category of authorship attribution approach builds upon neural network models [85, 28, 39]. In this approach, the sequence of words or characters are the input of a neural network which makes the proposed models utilize the sequential information. However, the proposed models in the literature mainly focus on lexical features despite the fact that lexical-based language models have very limited scalability when dealing with datasets containing diverse topics. On the other hand, syntactic models which are content-independent are more robust against topic variance. Zhang et. al. [100] introduces a strategy to incorporate syntactic information of documents in authorship attribution task. They propose a novel scheme to encode a syntax tree into a learnable distributed representation, and then integrate the syntax representation into a Convolutional Neural Network (CNN)-based model. Different from their approach, we are interested in a neural model which encodes the syntactic information without being equipped with explicit structural representation such as syntax parse tree. This is achieved by introducing a strategy to encode syntactic information of sentences using only their Part of Speech (POS) tags. We present a syntactic recurrent neural network which hierarchically learns and encodes the syntactic structure of documents. First, the syntactic representations of sentences are learned from the sequence of part-of-speech (POS) tags and then they are aggregated into document representations using recurrent neural networks. Afterwards, we use attention mechanism to highlight the sentences which contribute more to the detection of authorial writing style.

Furthermore, our motivation is to develop a neural model which preserves all the stylistic information of documents from all three levels of language production: lexical, syntactic, and structural. In the proposed model, we use lexical and syntactic embeddings to build two different sentence representations. These lexical and syntactic representations of sentences are independently fed into two parallel hierarchical neural networks to capture semantic and syntactic structure of sentences in documents. The hierarchical attention network captures the hierarchical structure of documents by constructing representations of sentences and aggregating them into document representations [98]. We employ convolutional layers as the word-level encoder to represent each sentence by its important lexical and syntactic n-grams, independent of their position in the sentence. For the sentence-level encoder, we employ an attention-based recurrent neural network to capture the structural patterns of sentences in the document. The primary reason for adopting recurrent architecture for sentence-encoder is because recurrent neural networks have been shown to be essential for capturing the underlying hierarchical structure of sequential data [94]. Ultimately, lexical and syntactic representations are fused and fed into a softmax classifier to predict the probability distribution over the class labels.

Additionally, we adopt a transfer learning approach and use deep contextualized word representation (ELMo) in our model to measure the impact of lower-level linguistic representations versus higher-level linguistic representations of ELMo in the task of authorship attribution. According to our experimental results, lower-level linguistic representations which mainly carry syntactic information, demonstrate better performance in authorship attribution task when compared to higher-level linguistic representations, which mainly carry semantic information.

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 [56], FastText [18], and Glove [63] 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 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 [46], InferSent [24], and Universal Sentence Encoder [22] are well-known examples.

There has been a growing interest in understanding what linguistic knowledge is encoded in deep contextual representations of language. For this purpose, several probing tasks are proposed to understand what these representations are capturing [93, 38, 25, 64]. One of the interesting findings is that, despite the existence of explicit syntactic annotations, these learned deep representations encode syntax to some extent [17]. Hewitt et. al. provide an evidence that the entire syntax tree is embedded implicitly in deep model's vector geometry. Kuncoro et. al. [50] 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 [84, 35], machine comprehension [54], summarization [87], text generation [12], machine translation [99, 51], authorship attribution [100, 41, 42], etc. Furthermore, Kuncoro et. al. provide evidence that models which have explicit syntactic information result in better performance [50]. 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 [41, 42]. We continue this work 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 [23] to explicitly learn the structural representation of sentences. The Siamese network is comprised 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. [25] 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.

Dissertation Organization

The rest of this dissertation is organized as follows:

Chapter 2 presents the literature review, which consists of two parts: Topic Detection and Authorship Attribution.

Chapter 3 presents our proposed model for topic detection from microblog streams.

Chapter 4 presents our proposed syntactic neural network and its application for the task of authorship attribution.

Chapter 5 presents our proposed style-aware neural model for the task of authorship attribution.

Chapter 6 elaborates our strategy to adopt a transfer learning of deep contextualized word embedding and fine-tune our model to measure the contribution of different linguistic features in the task of authorship attribution.

Chapter 7 presents our proposed self-supervised representation learning framework for learning the structural representation of sentence and its application in authorship attribution task.

Chapter 8 is the conclusion and future work.

CHAPTER 2: LITERATURE REVIEW

Topic Detection

General-purpose topic detection methods mainly fall into one of three classes: Feature Pivot Methods, Document Pivot Methods, and Probabilistic Models. Each of these three approaches has advantages and disadvantages. According to Fung. et al. [30], cluster fragmentation problem is one of the common drawbacks of document pivot methods, which leads to incorrect clustering. Probabilistic models usually produce good results; however, they are more computationally expensive. Feature pivot methods, based on analysis of terms correlation, often capture misleading term correlation due to noise in the data set. We discuss these three approaches in more detail in the following subsections.

Feature-pivot methods

Feature-pivot methods aim to find a group of terms which co-occur in a corpus. In these methods a topic is represented by a set of terms. Generally, feature-pivot methods include two steps towards detection of topics. First, a set of key terms is extracted from the corpus based on some importance measure and then the co-occurrence patterns between these key terms are computed. Second, these patterns are clustered based on some inter-term similarity measures, where each cluster represents a specific topic.

For instance, M. Cataldi et al. [21] consider both term frequency and also social features of tweets, like the popularity of the user, for selection of key terms. They utilize correlation vectors which represent the pairwise co-occurrence of terms in the corpus and generate a graph where each node identifies a term; the edge between two nodes represents the correlation vector of two terms. Finally, a graph-based algorithm is applied to generate the clusters.

J. Wang et al. [96] build frequency-based signals for individual terms and detects an event by grouping terms with similar patterns into a set. First, they select bursty terms by filtering away the trivial terms. Then for the clustering, a modularity-based graph partitioning is applied by computing the cross-correlation measures. H. Sayyadi et al [78] introduced a new event detection method which builds a keyword graph, *"KeyGraph"*, based on the probability of pairwise term co-occurances. The clustering method is a community detection algorithm which iteratively removes the edges with high betweenness. Regardless of the employed techniques for term selection and the clustering, most of the proposed methods attempt to examine pairwise correlation between terms. Considering correlation of more than two patterns are proposed as follows. J.Guo et al [33] treat the problem of topic detection as a Frequent Pattern Mining problem and propose a stream mining algorithm to detect topics from Twitter streams. Petkos et al [66] propose a softer version of FPM which represents the topic as frequent patterns.

Document-pivot methods

Document-pivot methods typically group together individual documents according to their similarity. The similarity measure is computed between either pairs of documents or a document and prototype cluster representation. In this approach a topic is represented by a set of documents. If the similarity of the incoming document is higher than some threshold, then the document is added to the cluster; otherwise a new cluster is created. The literature works which have adopted this approach mainly differ in the methods they applied to compute the similarity. For instance [68] compares the tf-idf vector of incoming tweets with the tf-idf vector of common terms in each cluster. In [75] a variant of incremental clustering is adopted in which the temporal and textual similarity of incoming tweets are considered. In this method the similarity between incoming news and the clusters older than some limit, or those which do not share any textual information, is not computed. This makes the method more appropriate for large databases. Another document-pivot approach is [67], which aims to address scalability issues by utilizing a modified version of LSH. In general, the document-pivot methods performance is dependent on the threshold parameter. These methods also suffer from the fragmentation issue, for which different merging procedures can be applied [14] [75].

Probabilistic Models

Probabilistic topic models deal with the distribution of topics and terms. In these approaches, the topic is represented as a distribution over terms. Latent Dirichlet Allocation (LDA) and Probabilistic Latent Semantic Analysis(PLSA) are two representative probabilistic topic models [40] [16] which have been extended widely. They use variables which represent per-topic term distribution and per-document topic distribution. In [71] the supervised version of LDA has been adopted for detecting topics and predicting links in Twitter. H. Kim et al [45] combined the frequent pattern mining method with probabilistic topic models and have reported performance improvements over LDA and PLSA for the classification task.

Authorship Attribution

In the field of Natural Language Processing, authorship analysis is the process of examining the linguistic style of a piece of text in order to draw some conclusions on its authorship. This subfield consists of three tasks; *authorship attribution* (to determine the likelihood of a particular author having written a given piece of text), *authorship profiling* (to detect characteristics of the author that has produced a given piece of text e.g. gender, age, personality,...) and *similarity* *detection* (to compare multiple pieces of work and decide if they have been written by a single author or not). Style-based text classification was introduced by Argamon-Engelson et al. [8]. The authors used basic stylistic features (the frequency of function words and part-of-speech 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 [31], forensics [6, 7, 73], social media analysis [13, 11, 37, 36], psycholinguistics [59, 62], and even source code analysis [5, 26]. Computational stylometry has been also explored in different languages including Chinese [83], Arabic [43, 34], Russian [53, 60], and Turkish [1], among others [44, 92].

Traditional Methods

In the traditional machine learning approach the stylistic features are engineered and extracted from the documents and are subsequently used as the inputs of different kinds of machine learning classifiers [81, 89, 82]. The engineered features reveal statistical information of documents in lexical, syntactic, and structural levels. Character/word n-grams, frequency of certain words and character distribution are examples of lexical-level features. The frequency of function words, punctuation, and syntactic n-grams are the most frequently used syntactic features. The number of words/sentences/paragraphs in a document and averaged word/sentence/paragraph length are instances of structural features.

Syntactic n-grams are shown to achieve promising results in different stylometric tasks including author profiling [70] and author verification [48]. 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 [72]. Sapkota and Bethard et al. propose a model that feeds affix and punctuation n-grams as syntactic features to an SVM classifier for the task of authorship attribution [76]. They demonstrate that character n-grams that capture information about affixes and punctuation play a key role in the predictions in the authorship attribution task. A combination of lexical and syntactic features has also been shown to enhance the model's 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 [90]. Furthermore, it has been shown that syntactic dependency and discourse features play a significant role in the task of gender and author identification and author verification [86]. Schwartz et al. combine lexical and syntactic features and use a linear classifier for writing style detection [80]. Kreutz et al. explore different ways to combine classifiers trained on lexical features (word n-grams) and syntactic features (PoS n-grams) to discriminate the language variety in written Dutch texts [49].

Deep Learning Based Methods

With recent advances in deep learning, there exists a large body of work in the literature which employs deep neural networks in the authorship attribution domain. In this approach, the sequence of words or characters are the input of a neural network which makes the proposed models utilize the sequential information. However, the proposed models in the literature mainly focus on lexical features despite the fact that lexical-based language models have very limited scalability when dealing with datasets containing diverse topics. On the other hand, syntactic models which are content-independent are more robust against topic variance.

Bagnall et al. have employed a recurrent neural network with a shared recurrent state which outperforms other proposed methods in PAN 2015 task [9]. 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 [32]. Ruder et al. proposes a hybrid CNN that leverages character and word sequences for the large-scale authorship attribution task. In their proposed model, characters are represented as one-hot vectors (discrete representations) [74]. In contrast to previous work, which uses discrete feature representations, Sari et al. presents a feed forward neural model which encodes each character n-gram as a continuous vector representation[77]. Shrestha et al. applies CNN based on character n-gram to identify the authors of tweets. Their proposed model takes a sequence of character n-grams as input. Despite the previous neural network models that use either a sequence of words or a sequence of characters as inputs, this model captures local interactions at the character-level. [85].

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 [39]. Zhang et al. introduces a syntax encoding approach using convolutional neural networks which combines with a lexical model and applies it to the domain of authorship attribution [100]. Their proposed approach utilized the syntax parse tree of sentences; however, we show that such an explicit annotation of hierarchical syntax is not necessary for the authorship attribution task. We propose a simpler and more effective way of encoding the syntactic information of documents for the domain of authorship attribution.

CHAPTER 3: MAXIMUM SEQUENCE MINING APPROACH FOR TOPIC DETECTION FROM MICROBLOG STREAMS

Topic Detection with Frequent Pattern Mining approach

[1] Mining frequent patterns in textual information for topic detection falls into the class of feature-pivot methods. Early feature-pivot methods in the literature examined the pairwise co-occurrence of terms. This approach suffers from producing mixed topics in heterogeneous streams where several stories are evolving in parallel. One of the solutions for dealing with this challenge is to take into account the co-occurrence of multiple terms rather than just a pair. Needless to say this approach will lead to higher quality results. The idea of exploiting Frequent Pattern Mining for detecting hot topics from Twitter was initiated by J. Guo et al. [33], who adopted an FP-stream algorithm in order to discover the patterns in Twitter streams, where a pattern is a set of terms which co-occur frequently.

G. Petkos et al. [66] proposed SFPM, a soft version of FPM, where a large number of terms in the patterns co-occur frequently rather than necessitating all the terms to appear. It is expected that using SFPM increases incorrectly correlated terms in the mining process, leading to lower keyword precision. L. Aiello et al. [2] compared six different topic detection methods and reported their corresponding keyword precision and recall. Inferred from the reported results, almost all methods have lower keyword precision than recall. This observation implies that most of the terms correlated incorrectly. Hence, we should use a mining algorithm that is able to capture the actual correlation of terms.

[1]The findings from this research has been previously published: Jafariakinabad, Fereshteh, and Kien A. Hua. ”Maximal sequence mining approach for topic detection from microblog streams.” 2016 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2016.

The relative positions of terms and the distance between the terms in the corpus can be employed as additional filters for the mining process. In order to capture the relative position of terms in the pattern, we propose to use a frequent sequence mining approach. Sequences are ordered lists of terms that are capable of capturing more semantic information. We adapt the Vertica mining of Maximal Sequential Patterns (VMSP) algorithm and propose a new text mining algorithm which aims to mine maximal sequences. Subsequently, we map the mined sequences into a directed graph and apply a community detection algorithm in order to cluster the patterns, where each cluster represents a specific topic. In the post-processing step, a set of key terms are selected for each cluster that represents the corresponding topic. In the following, we formulate the task of topic detection from microblogging streams and then describe our approach in details.

Mining of Maximal Sequences

Let text batch B_I be the set of all texts generated by a microbloging stream within a fixed time interval up to the time stamp I. If $T_i (i = 1, 2, \ldots, N)$ denotes the topic detected from batch B_I, B_I can be modeled as a set of multiple topics $T_I = \{T_1, T_2, \ldots, T_N\}$. The topic detection task in this study is defined as the task of detecting set T_I from the batch file B_I.

For mining purposes, we consider each user post as an individual transaction. Adopting this approach results in reducing the number of candidates for pattern generation, leading to a lower computational time. Moreover, it ultimately decreases the probability of generating mixed topics since it is unlikely to correlate terms in the different topics.

Definition 0 (Sequence). Let $T = \{t_1, t_2, \ldots, t_k\}$ be a set of terms. A sequence $S =< s_1, s_2, \ldots, s_n >$ $(s_i \in T)$ is an ordered list of terms. Each user post in the batch is a sequence of terms.

Definition 1 (Sub-sequence). Given a sequence $A =< a_1, a_2, \ldots, a_n >$ and a sequence $B =<$

$b_1, b_2, \ldots, b_m >$, sequence A is sub-sequence of B (denoted by $A \sqsubseteq B$) if there exist integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$, such that $a_1 = b_{i_1}, a_2 = b_{i_2}, \ldots, a_n = b_{i_n}$.

Definition 2 (Support). Given a batch file $B = \{S_1, S_2, \ldots, S_n\}$, where S_i is a sequence representing a transaction in B, $|B|$ is the number of posts in batch B. Let S be a sequence. We call S a sequence of B if there is a $S_i \in B$ such that $S \sqsubseteq S_i$. The support of S is the fraction of posts in batch B that contain S, denoted as $supp(S)$.

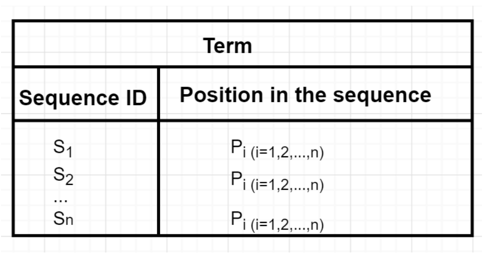
Definition 3 (Frequent Sequence). A sequence S is called frequent sequence if $supp(s)$ is greater than or equal to a user-predefined threshold, called the minimum support.

Definition 4 (Length of Sequence). The length of sequence S, denoted as $len(S)$, indicates the number of terms S contains.

In general, a frequent sequence mining algorithm may produce too many patterns. This problem not only makes the task of analyzing the pattern complicated and time consuming, but also demands more storage space [29]. A proper pruning scheme can be used in order to reduce the computational cost of the mining task and produce fewer but more representative patterns. Mining closed sequences and mining maximal sequences are two solutions for dealing with inherent redundancy of pattern mining algorithms. Sequences that are not included in any other sequence with the same support are denoted as closed sequences. A maximal sequence is a closed sequence which is not included in another closed sequence. Obviously, output space in the latter is smaller.

Definition 5 (Maximal Sequence). A frequent sequence S is a maximal sequence if there exist no frequent sequence S' such that $S \sqsubseteq S'$.

Definition 6 (Occurrence Database). An occurrence database is a database where each entry represents a term vector and indicates the list of sequences where the term appears along with the position of the term in the sequence. Figure 3.1 illustrates an entry of the occurrence database.



Term	
Sequence ID	Position in the sequence
S_1	$P_{i\ (i=1,2,\ldots,n)}$
S_2	$P_{i\ (i=1,2,\ldots,n)}$
...	
S_n	$P_{i\ (i=1,2,\ldots,n)}$

Figure 3.1: Term Vector: An entry of occurrence database

The initial step towards detecting topics from batch B_I is to find the set of maximal sequences S. Maximal sequence mining is substantial and useful in a wide range of applications; however, few algorithms have been proposed for this task since it is computationally expensive. We adapt the VMSP algorithm in order to discover all the frequent patterns in the batch. VMSP is one of the state-of-the-art algorithms for mining maximal sequences; by adopting a depth-first search method in the database and it is twice as fast as than the previously proposed algorithms [29]. The following text mining algorithm, Maximal Sequence Mining, is proposed to find maximal sequences from a corpus of text:

Algorithm 1 Maximal Sequence Mining

```
Input: B: A batch of tweets
       L: Maximum length of the sequence
       G: The gap between two terms in a sequence.
Output: Set of mined sequences
       Initialization:
1: Scan the batch of tweets to create the occurrence database and identify $S_{init}$, the list of frequent terms.
2: for each term $t$ in $S_{init}$ do
3:     Find $S_{sequels}$, the set of terms from $S_{init}$ which appears after $t$ in batch B.
4:     return FindMaximalSequence($t, S_{sequels}, L, G$)
5: end for
```

The MSM algorithm is actually finding the longest common subsequences in the corpus and it decreases redundancy of the mined patterns while preventing data loss. However, in most data

Algorithm 2 FindMaximalSequence()

```
1: for each term $t^{'}$ in $S_{sequels}$ do
2:     $S_{temp} = \emptyset$, $pattern$ = extension of $t$ with $t^{'}$
3:     if The extension of $t$ with $t^{'}$ is frequent and the length of $pattern$ is less than $L$ then
4:         $S_{temp} = S_{temp} \cup t$
           $S_{next} =$ Find $S_{sequels}$ the set of terms from $S_{init}$ which appears after $t$ in batch $B$
5:         return FindMaximalSequence($pattern, S_{next}, L, G$)
6:     end if
7: end for
```

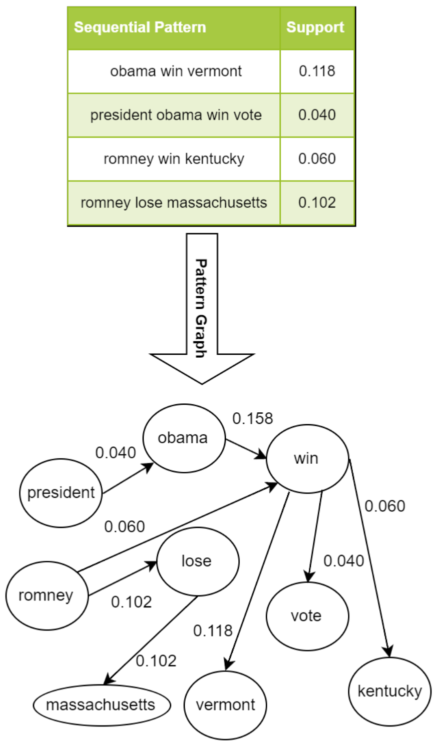
mining algorithms which adopt Apriori policy, long patterns tend not be mined due to the fact that it is less likely to be able to match patterns when the length of the pattern is long [97]. Therefore, long patterns are likely to encounter the low-frequency problem while using static minimum support for all patterns. In order to deal with this challenge we set minimum support very close to 0, which guarantees long patterns with low frequency will be mined. In order to examine the relative positional information of terms in the topic detection process, we also consider two parameters: maximum pattern length and maximum distance between terms.These parameters serve to control the strictness of the mining procedure.

Pattern Clustering

Each mined pattern holds some information about a certain topic. In order to generate the final topic, the patterns are clustered into groups where each group corresponds to a specific topic. Initially we map mined patterns into a directed graph (pattern graph) and then apply a community detection algorithm to cluster the patterns into different topics. In what follows, we describe each step in details.

Pattern Graph

A pattern graph is a directed graph in which each node represents a term and the edge between nodes indicates the co-occurrence of terms. Weight of the edge indicates pattern support and the direction implies the order in the pattern. In order to cluster the mined patterns, we first map the mined patterns into the pattern graph. Figure 3.2 illustrates the graph representation for some instances of mined maximal sequences.



Sequential Pattern	Support
obama win vermont	0.118
president obama win vote	0.040
romney win kentucky	0.060
romney lose massachusetts	0.102

Figure 3.2: An example of mapping mined sequences into a pattern graph

Pattern Clustering

We use community detection techniques in order to cluster the pattern graph. Generally, a community in a graph is a subgraph where the nodes are densely connected. Community detection algorithms, sometimes referred as graph partitioning methods, are aimed at dividing vertices of a graph into a number of communities[58]. C. Claudio et al. [19] adopts an edge removal approach for detecting communities, which finds the natural divisions of the vertices in a graph without requiring any input parameters, e.g. number of detected communities. The algorithm divides a graph into its subgraphs via iterative removal of the edges based on the edge clustering coefficient C_{ij}. C_{ij} is the fraction of the number of cycles that include a certain edge [19]: and is defined as

$$C_{ij}^{(g)} = \frac{z_{ij}^{(g)} + 1 * A_{ij}}{s_{i,j}^{(g)}}, \tag{3.1}$$

where $z_{ij}^{(g)}$ is the number of cycles of order g that includes the edge (i, j) with the weight of A_{ij} and $s_{i,j}^{(g)}$ is the number of possible cycles of order g in the given graph[19]. The underlying idea is that the edges between two different communities are unlikely to belong to many short loops. Therefore, inter community edges will have a low value of $C_{ij}^{(g)}$. After removing an inter community edge, the subgraph V is evaluated using the following definition of strong community [19]:

$$k_i^{in} > k_i^{out}, \forall i \in V, \tag{3.2}$$

where k_i^{in} is the number of edges that connect i to the nodes within V while k_i^{out} is the numebr of edges which connect i to the nodes in the rest of the graph[19]. After applying the algorithm, the graph will be divided into its subgraphs where the vertices are condensely correlated. Then each subgraph presents a set of terms, which co-occur frequently in the corpus.

Post-Processing

Each cluster generated by the previous step ideally includes all the patterns corresponding to a certain topic. The next step is to extract key terms for topic representation. We define a key node in a graph as a node that has the highest degree. In a graph the degree of a node is defined as follows:

$$D_i = k_i^{in} + k_i^{out} \tag{3.3}$$

Therefore, each cluster can be represented by a set of key vertices which hold highest amount of degree among all existing vertices in the subgraph. Ultimately, a topic is identified as a set of key terms.

Experimental Results

Our method was compared against four other methods: a document pivot method (LDA), a graph-based method, and two frequent pattern mining methods (FPM and SFPM). These methods were tested on three Twitter datasets containing real-world events in different domains. In the following, we first present the datasets and ground-truth data. Then we describe the evaluation method and data preprocessing procedure respectively. Ultimately, we present the experimental results.

Datasets

The experiments conducted in this study extract topics from three different datasets of tweets in the sport and political domains which were collected by L. Aiello et al.[2]. The datasets are collections of tweets related to three real-world events in 2012: FA cup final, U.S.A. elections, and Super Tuesday. Each collection is divided into different timeslots and the topics for all timeslots are known. The ground-truth topics include 22, 13, and 64 topics for Super Tuesday, FA cup, and USA Election datasets respectively. These topics are significant topics that are extracted manually and rely on mainstream media reports[2]. It is worth mentioning that the extracted topics are closely related, hence, the proposed datasets and the ground-truths are well-suited for examining the co-occurrence patterns of terms. Each topic is represented by the following sets of terms:

- Mandatory terms: these terms must appear in the candidate topic in order to be considered as correctly detected
- Optional terms: these terms may or may not appear in the topic
- Forbidden terms: these terms should not appear in the candidate topic. This set of terms is included in order to distinguish between closely connected topics.

Data Preprocessing

The preprocessing step plays an important role in the task of topic detection due to the high noisiness of user generated content, and involves data cleansing and noise removal. We use a preprocessing pipeline that includes the following steps:

- *Tokenization*: This step includes both sentence and word tokenization. A raw tweet is divided into a sequence of terms with hyperlinks, stop words, and punctuations removed. Hence, a sequence of cleaner terms is extracted from the raw posts.

- *lemmatization*: In information retrieval, stemming and lemmatization are used to reduce the feature space. Stemming is the task of reducing words to their stem while lemmatization aims to remove inflectional endings in order to return the base or dictionary form of a word, known as lemma. Lemmatization commonly collapses the different inflectional forms of a lemma while stemming most usually disintegrates derivationally related words. In this study, we use lemmatization as it is expected to perform more accurately than stemming. For example, stemming procedure of token *saw* might return just *s*, whereas lemmatization would attempt to return *see*.

In order to implement the preprocessing pipeline, we use CoreNLP toolkit [55] to extract clean and noise-free sets of term sequences from the raw datasets of tweets.

Evaluation

In order to evaluate MSM and compare it against different topic detection methods, we use an evaluation script proposed by L. Aiello et al.[2] where topic recall, keyword precision, and keyword recall are the reported evaluation metrics. According to the evaluation method, a topic is correctly detected if it contains all the mandatory terms and none of the forbidden terms. Topic recall is the fraction of ground-truth topics which are correctly detected. Keyword precision is the fraction of correctly detected keywords over the total number of keywords in the candidate topics that have been matched to some ground-truth topics. Keyword recall is the fraction of correctly detected keywords over the total number of keywords in the ground-truth topic that have been matched to some candidate topics. We added F-measure, which is the harmonic mean of keyword precision and keyword recall and it is suitable for measuring overall performance of the methods.

Note that topic precision, which is the fraction of detected topics over the total number of topics that took place at the specific time-slot, was not included in the evaluation. The reason behind is that there is no practical way to produce a definitive list of all topics in the batch, making it impossible to decide if a candidate topic is a real topic that took place in that time interval or not.

Table 3.1: Topic samples generated by MSM

Topic	**Topic relevant words in ground-truth**	**Detected Topical Terms**
	Super Tuesday	
Mitt Romney wins North Dakota	[mitt romney @mittromney];north;dakota;[win project call lead] cnn;ap	mitt,romney, win, dakota
Rick santorum makes a speech about healthcare	[rick santorum @ricksantorum];healthcare;speech	santorum,speech
	FA Cup	
Agger is shown yellow card for a tackle to Mikel	agger;[booked yellow card];tackle;mikel	mikel, yellow,agger,stoppage,chelsea
The final ends and chelsea wins liverpolll with 2-1	[final whistle gone]; chelsea; champions; congratulations; [2-1 2 1]; win	whistle, go ,chelsea, 2, liverpool ,1 ,final
	US Elections	
Obama wins Wisconsin	[barackobama barack obama]; [win call project held]; [wisconsin wi] cbs; fox	barackobama, win, first ,tweet
Jesse Jackson is re-elected in Chicago	[jesse jackson]; [wins re-elected re-election] ap; chicago; [rep representatives]	barackobama , win

These measures are computed for the top N topics produced by the detection algorithms. The final performance measure for a dataset is the micro-average of measures corresponding to all time-slots in the dataset. Table 3.1 shows examples of ground-truth topics and also topics detected using the proposed method.

Parameter Tuning

In this part, we examine the effect of different parameters on the performance of MSM. Owing to space limitations, we only demonstrate the performance measure results tested on the Super Tuesday dataset. The performance metrics show similar behavior in three different datasets. Figure 3.3 demonstrates the keyword precision, keyword recall, and keyword F-measure across different values of maximum pattern length (L) and maximum distance between terms (Gap) where the

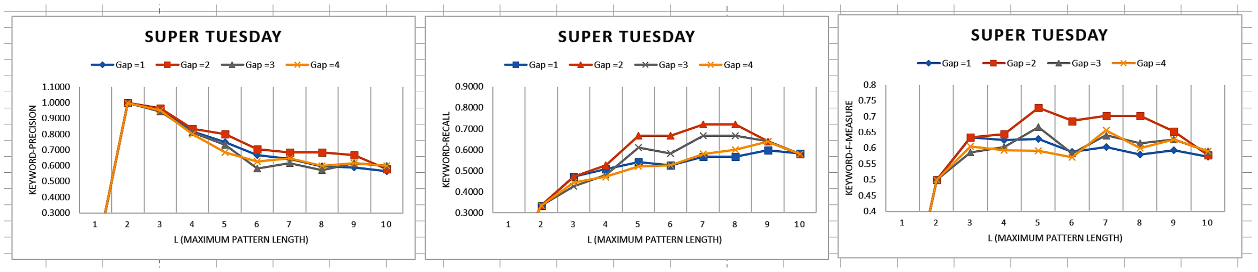


Figure 3.3: Keyword performance measures across different values of L and Gap in the Super Tuesday dataset

minimum support for mining sequences is set to 0.01(minimum support is explained in Section 3). It can be inferred from the charts that precision decreases when increasing L, because longer patterns are more likely to be wrongly correlated terms, causing the detected topics to contain more unrelated terms to the real topics. However, the figure demonstrates that keyword-recall grows when increasing the maximum pattern length. The reason is that the longer the mined patterns are, the more information is revealed about the real topics. To see the overall effect of maximum pattern length on the performance of MSM, F-measure is the metric to observe. We can observe from the charts in the figure that F-measure initially grows and then decreases when enlarging L, and generally peaks when L is set to 5. This is expected since F-measure is a trade off between precision and recall.

On the other hand, Gap parameter shows similar behavior. It can be observed from the mentioned figures that all three metrics including precision, recall and F-measure initially increase and then decrease when increasing Gap. A low value of Gap indicates more strictness of the algorithm in grouping terms, causing mined patterns to hold contiguous terms. However, when Gap is set to a larger value, MSM will also group terms which are not strongly correlated.

Figure 3.4 illustrates the topic recall performance of three Twitter datasets across different values of

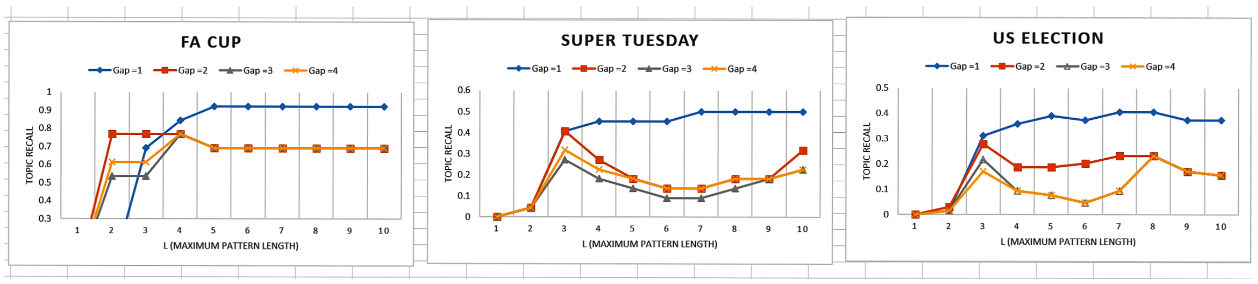


Figure 3.4: Topic Recall measures of different values of L and Gap for three Twitter datasets

Gap and L. According to the figure, topic recall initially grows when enlarging L, then decreases, and finally trends towards stability when L is enlarged to a certain number e.g. 10. The reason behind this is that patterns with the high value of L provide more information about the target topic; however, longer patterns are more probable to wrongly associate terms. Additionally, topic recall decreases when increasing Gap, since the low value of Gap yields to extracting strongly correlated terms. According to the observations from the figures, MSM approach shows its highest topic recall performance when Gap and L are set to 1 and 5 respectively.

Results

Table 3.2 shows the evaluation results of topic detection methods for the top N detected topics. For US Elections and Super Tuesday, the top 10 detected topics are considered for evaluations; however, in FA Cup, due to the smaller number of topics and shorter timeslots, the top 2 detected topics are used. Performance evaluation of LDA, Graph-based, FPM and SFPM are reported by G. Petkos et al. [66]. We use the same datasets and same open source evaluation script[2]. T-Recall, K-Recall, and K-Precision refer to topic recall, keyword recall, and keyword precision respectively.

[2]http://www.socialsensor.eu/results/datasets/72-twitter-tdt-dataset

According to the results, MSM approach significantly outperforms the other methods with its best topic recall score for all three datasets. Moreover, it performs well in keyword precision. This indicates that, as we expected, the MSM approach captures more accurate term correlations due to the use of relative positional information as a filter in the mining process. Although MSM approach performs less in terms of keyword recall, overall it achieves the highest performance in keyword F-measure compared to the other methods in the table. Therefore, MSM approach is able to detect more topics and represent a topic in a more accurate manner. SFPM shows higher performance in keyword recall because it is not as strict as MSM in grouping terms; hence, it clearly correlates more terms. Using more accurate methods for selecting key terms from a topic cluster may improve the performance of the MSM approach in keyword recall.

Table 3.2: Performance of topic detection methods in three Twitter datasets

Super Tuesday

Method	T-Recall	K-Precision	K-Recall	F-Measure
LDA	0.0000	0.0000	0.0000	0.0000
Graph-based	0.0455	0.3750	0.6000	0.4615
FPM	0.1364	1.0000	0.4091	0.5806
SFPM	0.1818	0.4717	**0.8929**	0.6117
MSM	**0.4550**	**0.7500**	0.5410	**0.6285**

US Elections

Method	T-Recall	K-Precision	K-Recall	F-Measure
LDA	0.1094	0.1654	0.6286	0.2618
Graph-based	0.0781	0.3750	0.4839	0.4225
FPM	0.0000	0.0000	0.0000	0.0000
SFPM	0.3594	0.2412	**0.6953**	0.3581
MSM	**0.3910**	**0.6150**	0.5400	**0.5751**

FA Cup

Method	T-Recall	K-Precision	K-Recall	F-Measure
LDA	0.6923	0.6585	0.1578	0.2545
Graph-based	0.2307	0.4285	0.2857	0.3428
FPM	0.6923	0.6428	0.2967	0.4060
SFPM	0.9230	**0.6666**	0.2186	0.3292
MSM	**0.9230**	0.6120	**0.5560**	**0.5826**

Summary

In this chapter, we presented a Maximum Sequence Mining (MSM) approach, which is a feature-pivot topic detection method that examines the co-occurrence patterns of terms in the corpus. Its novelty lies in the patterns used for the mining process that are sequences of terms, as opposed to sets of terms without any particular order. The former pattern representation captures the positional information of the terms in the sequence and is more accurate in reflecting the semantics of the underlying content. Based on this sequence concept, a MSM algorithm is introduced to compute the frequent sequences from a batch of social streams. A directed-graph representation of these sequences, called pattern graph, can then be constructed; and a community detection algorithm is used to partition the nodes in the pattern graph into clusters, each corresponding to a distinct topic. Each topic is represented by a set of keywords selected from the corresponding cluster. Our experiments indicate that the proposed technique performs well in keyword precision. Although it performs less in terms of keyword recall, overall it outperforms current state-of-the-art techniques in topic detection with its superior topic recall score.

CHAPTER 4: SYNTACTIC NEURAL MODEL FOR AUTHORSHIP ATTRIBUTION

[1] We introduce a syntactic neural model to encode the syntactic patterns of a document in a hierarchical structure. First, we represent each sentence as a sequence of part-of-speech (POS) tags. Each POS tag is embedded into a low dimensional vector which is fed into a POS encoder which learns the syntactic representation of sentences. Subsequently, the learned sentence representations are aggregated into the document representation. Moreover, we use an attention mechanism to highlight the sentences which contribute more to the prediction of labels. Finally, we use a softmax classifier to compute the probability distribution over class labels. The overall architecture of the network is shown in figure 4.1. In the following sections, we elaborate on the main components of the model.

POS Embedding

We assume that each document is a sequence of M sentences and each sentence is a sequence of N words, where M and N are model hyperparameters and the best values of which are explored through the hyperparameter tuning phase. Given a sentence, we convert each word into the corresponding POS tag in the sentence and afterwards we embed each POS tag into a low dimensional vector $P_i \in R^{d_p}$ using a trainable lookup table $\theta_P \in R^{|T| \times d_p}$, where T is the set of all possible POS tags in the language. We use NLTK part-of-speech tagger [15] for the tagging purpose and use the

[1]The findings from this research has been previously published: Jafariakinabad, Fereshteh, Sansiri Tarnpradab, and Kien A. Hua. "Syntactic Neural Model for Authorship Attribution." The Thirty-Third International Flairs Conference. 2020.

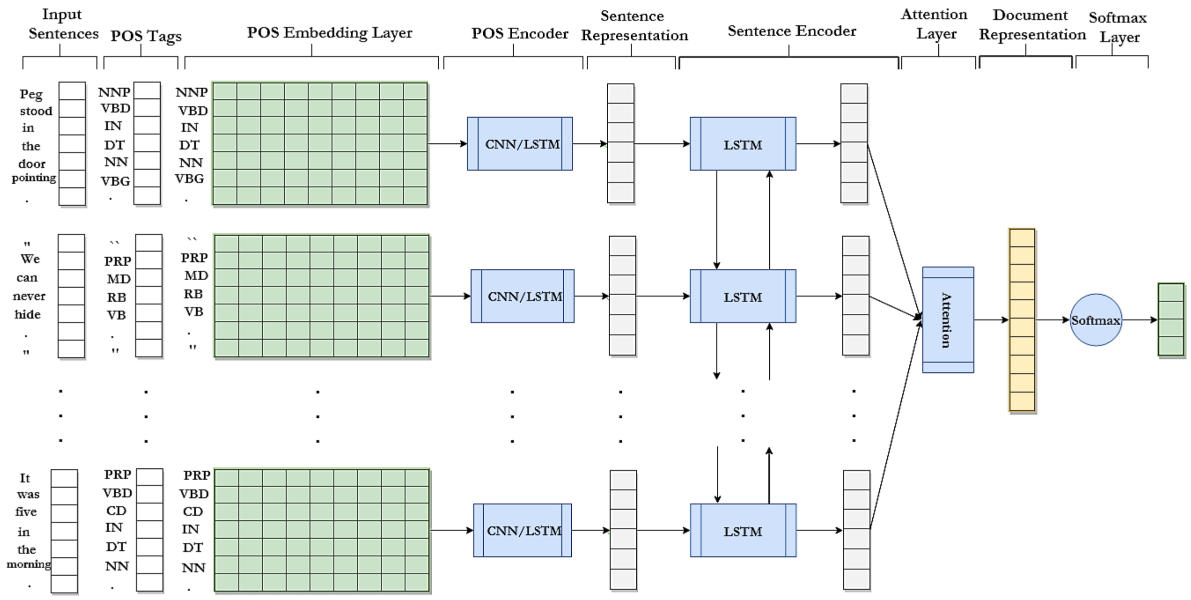


Figure 4.1: The Overall Architecture of Syntactic Neural Model

set of 47 POS tags[2] in our model as follows.

T = { *CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, LS, MD, NN, NNS, NNP, NNPS, PDT, POS, PRP, PRP$, RB, RBR, RBS, RP, SYM, TO, UH, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WP$, WRB, ‘,’, ‘:’, ‘...’, ‘;’, ‘?’, ‘!’, ‘.’, ‘$’, ‘(’, ‘)’, “‘ ’, ‘” ’*}

One of the advantages of using POS tags instead of words is its low dimensional lookup table, compared to the word embeddings where the size of vocabulary in large datasets usually surpasses 50,000 words. Compared to word embeddings, the size of POS embedding lookup table is significantly smaller, fixed, and independent of the dataset which makes the proposed model less likely to have out-of-vocabulary words.

[2]`https://github.com/nltk/nltk/blob/develop/nltk/app/chunkparser_app.py`

POS Encoder

POS encoder learns the syntactic representation of sentences from the output of POS embedding layer. In order to investigate the effect of short-term and long-term dependencies of POS tags in the sentence, we exploit both CNNs and LSTMs.

Short-term Dependencies

Let $S_i = [P_1; P_2; ...; P_N]$ be the vector representation of sentence i and $W \in R^{rd_p}$ be the convolutional filter with receptive field size of r. We apply a single layer of convolving filters with varying window sizes as the of rectified linear unit function (ReLU) with a bias term b, followed by a temporal max-pooling layer which returns only the maximum value of each feature map $C_i^r \in R^{N-r+1}$. Consequently, each sentence is represented by its most important syntactic n-grams, independent of their position in the sentence. Variable receptive field sizes Z are used to compute vectors for different n-grams in parallel and they are concatenated into a final feature vector $h_i \in R^K$ afterwards, where K is the total number of filters:

$$C_{ij}^r = relu(W^T S_{j:j+r-1} + b), j \in [1, N-r+1],$$

$$\hat{C_i^r} = max\{C_i^r\},$$

$$h_i = \oplus \hat{C_i^r}, \forall r \in Z$$

Long-term Dependencies

Let $S_i = [P_1; P_2; ...; P_N]$ be the vector representation of sentence i. As an alternative to CNN, we use a bidirectional LSTM to encode each sentence. The forward LSTM reads the sentence S_i from P_1 to P_N and the backward LSTM reads the sentence from P_N to P_1. The feature vector $h_t^p \in R^{2d_l}$ is the concatenation of the forward LSTM and the backward LSTM, where d_l is the dimensionality of the hidden state. The final vector representation of sentence i, $h_i^s \in R^{2d_l}$ is computed as the unweighted sum of the learned vector representation of POS tags in the sentence. This allows us to represent a sentence by its overall syntactic pattern.

$$\overrightarrow{h_t^p} = LSTM(P_t), t \in [1, N],$$

$$\overleftarrow{h_t^p} = LSTM(P_t), t \in [N, 1],$$

$$h_t^p = [\overrightarrow{h_t^p}; \overleftarrow{h_t^p}]$$

$$h_i^s = \sum_{t \in [1,N]} h_t^p$$

Sentence Encoder

Sentence encoder learns the syntactic representation of a document from the sequence of sentence representations outputted from the POS encoder. We use a bidirectional LSTM to encode the sentences of which the outputted vector is calculated as follows.

$$\overrightarrow{h_i^d} = LSTM(h_i^s), i \in [1, M],$$

$$\overleftarrow{h_i^d} = LSTM(h_i^s), i \in [M, 1],$$

$$h_i^d = [\overrightarrow{h_i^d}; \overleftarrow{h_i^d}]$$

Clearly, not all sentences are equally informative about the authorial style of a document. Therefore, we incorporate an attention mechanism to reveal the sentences that contribute more to detection of the writing style. We define a sentence level vector u_s and use it to measure the importance of the sentence i as follows:

$$u_i = tanh(W_s h_i^d + b_s)$$

$$\alpha_i = \frac{exp(u_i^T u_s)}{\sum_i exp(u_i^T u_s)}$$

$$V = \sum_i \alpha_i h_i^d$$

Where u_s is a learnable vector and is randomly initialized during the training process. V is the vector representation of the document, which is the weighted sum of vector representations of all sentences.

Classification

The learned vector representations of documents are fed into a softmax classifier to compute the probability distribution of class labels. Suppose V_k is the vector representation of document k learned by the attention layer. The prediction $\tilde{y_k}$ is the output of the softmax layer and is computed as:

$$\tilde{y_k} = softmax(W_c V_k + b_c)$$

Where W_c, and b_c are learnable weight and learnable bias, respectively, and $\tilde{y_i}$ is a C dimensional vector (C is the number of classes). We use cross-entropy loss to measure the discrepancy of predictions and true labels y_k. The model parameters are optimized to minimize the cross-entropy loss over all the documents in the training corpus. Hence, the regularized loss function over N documents denoted by $J(\theta)$ is:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} log \tilde{y}_{ik} + \lambda||\theta||$$

Experimental Results

Dataset

We evaluate our proposed method on a commonly used benchmark dataset from PAN 2012 authorship attribution shared task[3]. We chose Task I dataset which corresponds to the authorship attribution among a closed set of 14 authors. The training set comprises 28 novel-length documents (two per candidate author), ranging from 32,000 words up to approximately 180,000 words. The test set consists of 14 novels (one per candidate author) with the length ranging from 42,000 words up to 190,000 words. Table 4.1 reports the word count and the averaged sentence length of documents in both train and test set for each candidate author(The numbers are rounded down).

In order to generate enough train/test samples, we have schematized the novels into the segments with a M number of sentences (sequence length). The best value of M is explored through the hyperparameter tuning phase. Accordingly, the performance measures include segment-level categorical accuracy as well as document-level categorical accuracy. In the latter, we use majority voting to label a document based on the segment-level predictions.

[3]`https://pan.webis.de/clef12/pan12-web/authorship-attribution.html`

Table 4.1: Corpus Statistics

	Training Data I		Training Data II		Test Data	
	Word Count	Sentence Length	Word Count	Sentence Length	Word Count	Sentence Length
Candidate 01	73,449	17	76,602	19	70,112	20
Candidate 02	180,660	13	117,024	14	82,317	13
Candidate 03	158,306	17	121,301	19	151,049	15
Candidate 04	84,080	14	79,413	18	93,055	14
Candidate 05	109,857	18	141,086	15	96,663	15
Candidate 06	61,644	19	46,549	16	42,808	16
Candidate 07	71,106	16	70,563	18	84,996	21
Candidate 08	106,024	18	113,475	15	94,700	13
Candidate 09	66,840	15	41,093	15	194,547	15
Candidate 10	86,681	14	35,699	16	60,998	16
Candidate 11	53,960	19	48,037	13	80,330	24
Candidate 12	49,543	25	64,495	26	50,636	27
Candidate 13	32,900	21	153,994	32	77,780	27
Candidate 14	89,908	23	71,058	22	52,633	35

Baselines

For our baselines, we employ standard syntactic n-gram model as a syntactic approach and word n-gram model as a lexical approach. For both models, we have used Support Vector Machine (SVM) classifier with linear kernel. Moreover, in order to compare the performance of syntactic recurrent neural network to the lexical based approaches, we fed the sequence of words to a neural network with identical architecture. We use 300-dimensional pretrained Glove embeddings [63] for the embedding layer in the network.

Hyperparameter Tuning

In this section, we examine the effect of different hyperparameters on the performance of the proposed model. All the performance metrics are the mean of segment-level accuracy (on the test set) calculated over 10 runs with 0.9/0.1 train/validation split. We use Nadam optimizer [91] to optimize the cross-entropy loss over 30 epochs of training.

CNN for POS encoding

Figure 4.2 illustrates the performance of the syntactic recurrent neural network when CNN is used as POS encoder, across different receptive field sizes and number of layers, while other parameters are kept constant. We observe that increasing the number of convolutional layers generally lessens the performance. Moreover, in one convolutional layer, the accuracy generally increases by increasing the size of receptive fields. This can be due to the fact that receptive fields with the larger sizes capture longer syntactic sequences which are more informative.

In our experiments, we also observed that having parallel convolutional layers with different receptive field sizes improves the performance. Therefore, in the final model, we use one layer of multiple convolutional filters with the receptive field sizes of 3 and 5.

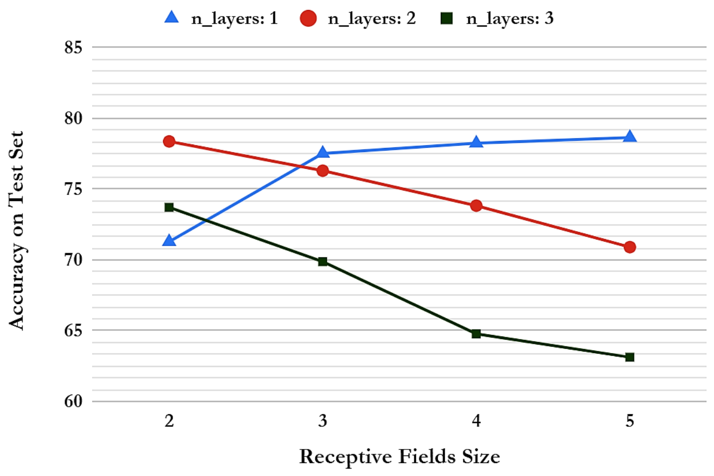


Figure 4.2: The performance of syntactic model across different receptive field sizes and number of layers(n_layers)

LSTM for POS encoding

Figure 4.3 demonstrates the accuracy of the proposed model when LSTM is employed as POS encoder, across different values of sentence length (N) and sequence length (M: the number of sentences in each segment). We observe from the figure that increasing the sequence length boosts the performance and the model achieves higher accuracy on the segments with 100 sentences (74.40%) than the segments with only 20 sentences (60.02%). This observation confirms that the investigation of writing style in short documents is more challenging [57].

As shown in Table 4.1, the average sentence length in the dataset ranges from 13 to 35. Therefore, we have examined the sentence length of 10, 20, 30, and 40 (the performance of the model is identical when the sentence length is 30 and 40, so we have not included the latter results in the figure). We observe that increasing the length of sentences to 30 words improves the performance, primarily because decreasing the sentence length ignores several words in the sentence which leads to notable information loss. To sum up, the syntactic neural network accepts segments as the inputs where each segment contains 100 sentences and the length of each sentence is 30.

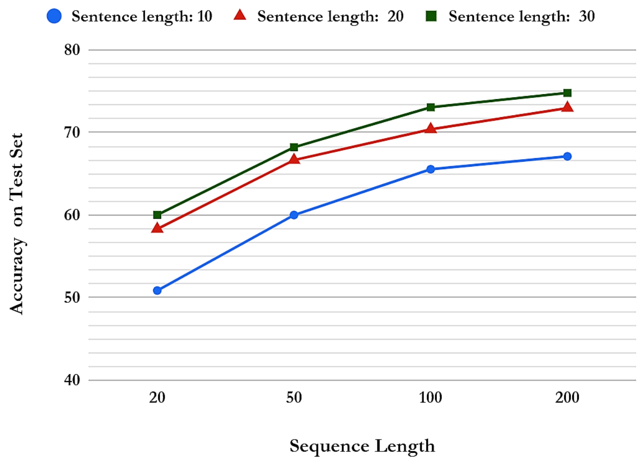


Figure 4.3: The performance of syntactic model across different sentence lengths and sequence lengths

Table 4.2: The performance results of models on PAN 2012 dataset for authorship attribution task

Model		**Segment-Level Accuracy (%)**		**Document-Level Accuracy(%)**
		Validation	**Test**	
Lexical	Word N-grams-SVM	90.71	58.35	78.57 (11/14 novels)
	CNN-LSTM	**98.88**	64.12	78.57 (11/14 novels)
	LSTM-LSTM	96.83	63.92	85.71 (12/14 novels)
Syntactic	POS N-grams-SVM	89.60	69.66	92.85 (13/14 novels)
	CNN-LSTM	93.22	**78.76**	**100.00** (14/14 novels)
	LSTM-LSTM	95.00	74.40	**100.00** (14/14 novels)

Results

We report both segment-level and document-level accuracy. As mentioned before, each document (novel) has been divided into the segments of 100 sentences. Therefore, each segment in a novel has been classified independently and afterwards the label of each document is calculated as the majority voting of its constituent segments. Table 4.2 reports the performance results of baselines and the proposed model (with both CNN and LSTM as POS encoder) on the PAN 2012 dataset. According to the segment-level accuracy, the performance of all models has dropped significantly on the test set, mainly because of insufficient training data. We expect that if the models are trained on enough writing samples per author, the test results would be closer to the validation results.

Unsurprisingly, the syntactic CNN-LSTM model outperforms the conventional POS n-gram model (POS N-gram-SVM) by 9.1% improvement in segment-level accuracy and 7.15% improvement in document-level accuracy. This is primarily because syntactic CNN-LSTM not only represents a sentence by its important syntactic n-grams but also learns how these sentences are structured in a document. On the other hand, the POS N-gram-SVM model only captures the frequency of different n-grams in the document.

Syntactic v.s. Lexical

According to Table 4.2, both syntactic recurrent neural networks (CNN-LSTM and LSTM-LSTM) outperform the lexical models by achieving the highest document-level accuracy (100.00%). Syntactic recurrent neural networks have correctly classified all the 14 novels in the test set while lexical LSTM-LSTM achieves the highest document-level accuracy (85.71%) in the lexical models by correctly classifying 12 novels.

In segment-level classification, syntactic recurrent neural networks outperform the lexical models in the test set with 14% higher accuracy; however, the lexical models achieve higher validation accuracy. This observation may imply the lower generalization capability of lexical models compared to the syntactic models in the style-based text classification.

Short-Term v.s. Long-Term

According to the results in table 4.2, syntactic CNN-LSTM model slightly outperforms syntactic LSTM-LSTM by approximately 4% in segment-level accuracy. The primary difference between the two models is the way they represent a sentence. In syntactic CNN-LSTM, each sentence is represented by its important syntactic n-gram independent of their position in the sentence. However, syntactic LSTM-LSTM mainly captures the overall syntactic pattern of a sentence by summing up all the learned vector representations of POS tags in the sentence.

Short Documents v.s. Long Documents

We have conducted a controlled study on the effect of document length on the performance of both CNN-LSTM and LSTM-LSTM models. For this purpose, we have trained each model on only a specific fraction of each training document and afterwards tested the trained model on the whole test set. We keep the number of model parameters in both models approximately equal to eliminate the effect of data limitation on the training process. Figure 4.4 demonstrates the performance results of models when trained on the first $n\%$ of segments in each document. For example when n is equal to 10, it means the models are trained on only the first 10% of segments in the documents rather than the whole.

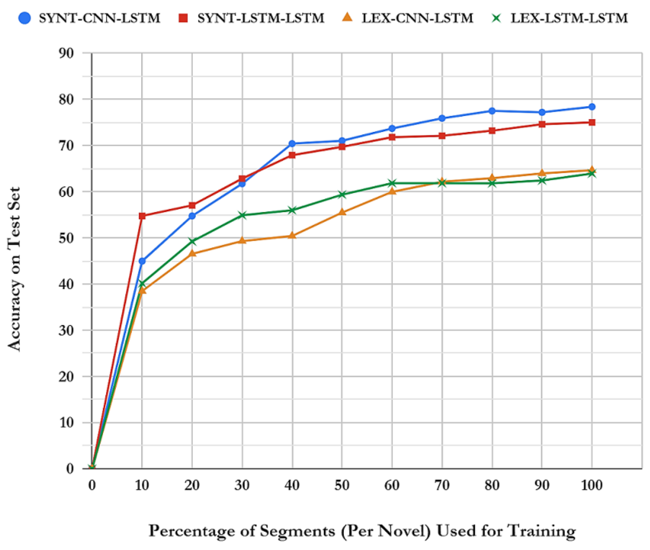


Figure 4.4: The performance of syntactic models when trained on different number of segments per novel

We observe that when the smaller portion of segments ($< 30\%$) are used for training, LSTM-LSTM models achieve higher test accuracy than CNN-LSTM models in both syntactic and lexical settings. On the other hand, CNN-LSTM models slightly outperform LSTM-LSTM models when the number of segments used for training in each document increases. In other words, LSTM-LSTM models appear to be quicker in capturing authorial writing style than CNN-LSTM models. This property, in particular, makes them a preferred potential model when investigating authorial writing style in a dataset of short documents.

Summary

In this chapter, we presented a syntactic neural model in order to encode the syntactic patterns of documents in a hierarchical structure and afterwards used the learned syntactic representation of documents for the authorship attribution task. We investigated both long-term and short-term dependencies of part-of-speech tags in sentences. According to our experimental results on PAN 2012 dataset, syntactic models outperform lexical-based models by 14% in terms of segment-level accuracy. Moreover, we observed that LSTM-based POS encoders are quicker in capturing the authorial writing style than CNN-based POS encoders. This property makes them a preferable model when investigating authorial writing style in a dataset of short documents.

CHAPTER 5: STYLE-AWARE NEURAL MODEL FOR AUTHORSHIP ATTRIBUTION

[1]

We introduce a neural network which encodes the stylistic information of documents from three levels of language production (lexical, syntactic, and structural). We assume that each document is a sequence of M sentences and each sentence is a sequence of N words, where M and N are model hyperparameters and the best values are explored through the hyperparameter tuning phase (Section 5). First, we obtain both lexical and syntactic representation of words using lexical and syntactic embeddings respectively. These two representations are fed into two identical hierarchical neural networks which encode the lexical and syntactic patterns of the 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.

Employing both lexical and syntactic information of documents, as well as using the hierarchical neural network to encode the semantic and syntactic structure of sentences in documents, leads to a neural network which is capable of capturing main stylistic features. The hierarchical neural network is comprised of convolutional layers as a word-level encoder to obtain the sentence representations. They are then aggregated into document representation using recurrent neural networks. Finally, we use an attention mechanism to reward the sentences which contribute more to the detection of authorial writing style. The overall architecture of the proposed model is shown

[1]The findings from this research has been previously published: Jafariakinabad, Fereshteh, and Kien A. Hua. "Style-aware neural model with application in authorship attribution." 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 2019 and Jafariakinabad, Fereshteh, and Kien A. Hua. "Unifying Lexical, Syntactic, and Structural Representations of Written Language for Authorship Attribution." SN Computer Science 2.6 (2021): 1-14.

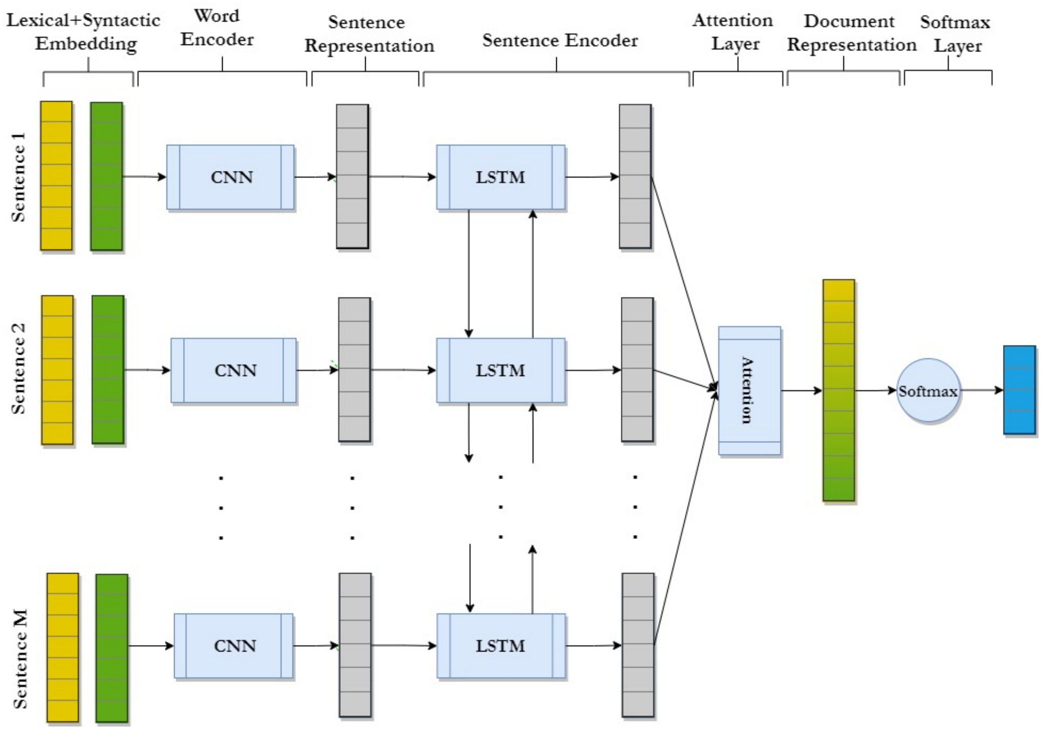


Figure 5.1: The overall architecture of Style-Aware Neural Model

in figure 5.1. We elaborate each component in the following subsections.

Lexical and Syntax Encoding

We encode the semantic and syntactic information of documents independently using lexical and syntactic embeddings, which is illustrated in figure 7.7. These two representations will feed into two parallel hierarchical networks. Hence, the syntactic and semantic patterns of document are learned independently from each other.

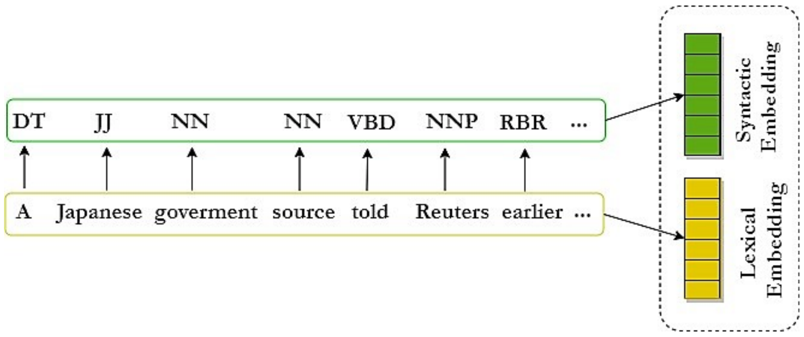

DT JJ NN NN VBD NNP RBR ...
A Japanese goverment source told Reuters earlier ...
Syntactic Embedding
Lexical Embedding

Figure 5.2: Lexical and Syntactic Embedding

Lexical Embedding

In the lexical-level we embed each word to a vector representation. We use pre-trained Glove embeddings [63] and represent each sentence as the sequence of its corresponding word embeddings.

Syntactic Embedding

Syntactic features play an important role in the domain of authorship attribution since they capture the syntactic patterns adopted by a specific author and reveal how the author structures the sentences in a document. This family of stylistic features is largely disregarded in the proposed deep learning based models in the literature and they mostly focus on lexical features. Even though word embeddings implicitly capture syntactic information of sentences, explicit modeling of the syntactic structure of sentences has been shown to further improve the performance of classifiers in different NLP tasks, including machine translation [54, 51], document summerization [87], text generation [12], and authorship attribution [100]. Inspired by these observations, we propose a simple way to encode the syntactic information of sentences for the domain of authorship attribution as the following.

Given a sentence, we convert each word into the corresponding POS tag in the sentence, and then embed each POS tag into a low dimensional vector $P_i \in R^{d_p}$ using a trainable lookup table $\theta_P \in R^{|T| \times d_p}$, where T is the set of all possible POS tags in the language. We use NLTK part-of-speech tagger [15] for the tagging purpose and use the set of 47 POS tags[2] in our model as follows.

T = { *CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, LS, MD, NN, NNS, NNP, NNPS, PDT, POS, PRP, PRP$, RB, RBR, RBS, RP, SYM, TO, UH, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WP$, WRB, ‘,’, ‘:’, ‘...’, ‘;’, ‘?’, ‘!’, ‘.’, ‘$’, ‘(’, ‘)’, “‘ ’, ‘” ’*}

Hierarchical Model

Word-level Encoder

The outputs of the lexical and syntactic embedding layers go into two identical convolutional layers (lexical-CNN and Syntactic-CNN) which learn the semantic and syntactic patterns of sentences in parallel. Due to the identical architecture of both networks, we only elaborate on the Syntactic-CNN in what follows. Figure 5.3 illustrates the architecture of Syntactic-CNN.

Let $S_i = [P_1; P_2; ...; P_N]$ be the vector representation of sentence i (P_i denotes the vector representation of POS tag i from the embedding layer), and $W \in R^{rd_p}$ be the convolutional filter with receptive field size of r. We apply a single layer of convolving filters with varying receptive field sizes and the rectified linear unit function (relu) as the activation function with a bias term b. Then a temporal max-pooling layer is applied which returns only the maximum value of each feature

[2]https://github.com/nltk/nltk/blob/develop/nltk/app/chunkparser_app.py

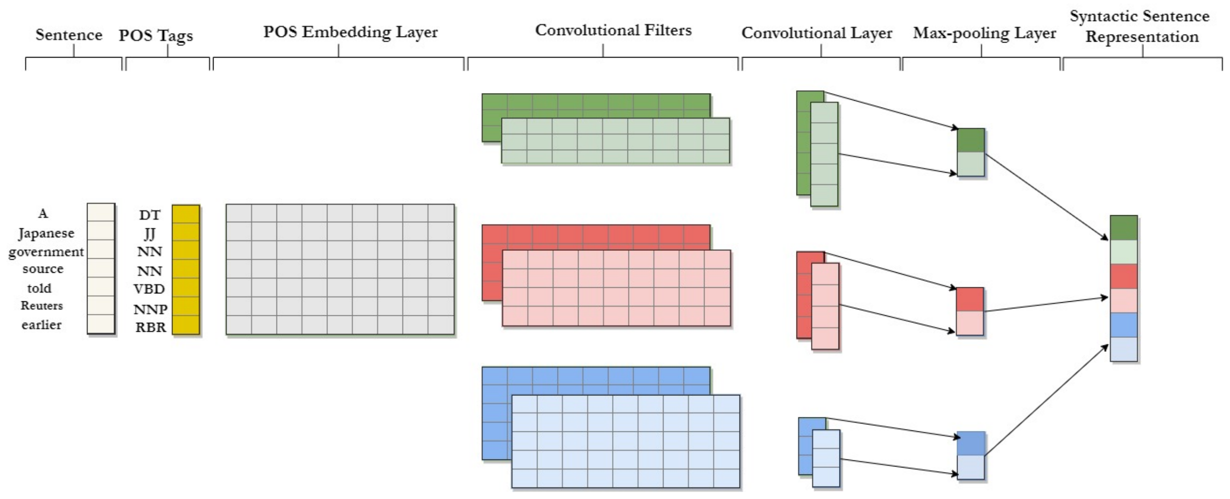


Figure 5.3: Syntactic-CNN having three parallel convolutional layers with receptive field sizes of 3,4, and 5.

map $C_i^r \in R^{N-r+1}$. Each sentence is then represented by its most important syntactic n-grams, independent of their position in the sentence. Variable receptive field sizes Z are used to compute vectors for different n-grams in parallel; and they are concatenated into a final feature vector $h_i \in R^K$ afterwards, where K is the total number of filters:

$$C_{ij}^r = relu(W^T S_{j:j+r-1} + b), j \in [1, N-r+1],$$

$$\hat{C}_i^r = max\{C_i^r\},$$

$$h_i = \oplus \hat{C}_i^r, \forall r \in Z$$

Sentence-level Encoder

The sentence encoder learns the lexical/syntactic representation of a document from the sequence of sentence representations output from the word-level encoder. Let h_i^s be the output of the word-level encoder where i denotes the index of the sentence in the document. We use a bidirectional LSTM to capture how sentences with different syntactic patterns are structured in a document. The vector output from the sentence encoder (h_i^d) is the concatenation of forward LSTM ($\overrightarrow{h_i^d}$) and backward LSTM ($\overleftarrow{h_i^d}$) and is calculated as follows. In the equations, M denotes the total number of sentences in the document.

$$\overrightarrow{h_i^d} = LSTM(h_i^s), i \in [1, M],$$

$$\overleftarrow{h_i^d} = LSTM(h_i^s), i \in [M, 1],$$

$$h_i^d = [\overrightarrow{h_i^d}; \overleftarrow{h_i^d}]$$

Clearly, not all sentences are equally informative about the authorial style of a document. Therefore, we incorporate an attention mechanism to reward the sentences that contribute more in detecting the writing style. First, we feed the sentence vector h_i^d through a one-layer MLP with $tanh$ activation to get u_i as a hidden representation of h_i^d. Then we define a sentence level vector u_s to measure the importance of the sentence i and get a normalized importance weight α_i through a softmax function as follows:

$$u_i = tanh(W_s h_i^d + b_s)$$

$$\alpha_i = \frac{exp(u_i^T u_s)}{\sum_i exp(u_i^T u_s)}$$

$$V = \sum_i \alpha_i h_i^d$$

Where W_s and b_s are the learnable weight and learnable bias of the MLP layer, respectively. u_s is a learnable vector and is randomly initialized during the training process; V is the vector representation of the document, which is the weighted sum of the vector representations of all sentences.

The primary reason for adopting recurrent architecture for sentence-encoder is because recurrent neural networks have been shown to be essential for capturing the underlying hierarchical structure of sequential data [94]. By adopting this approach, sentence-encoder is able to encode how sentences are structured in a document. Accordingly, structural information of documents are incorporated into the final document representation.

Lexical and Syntactic Representations Fusion

In this phase, the semantic and syntactic representations of the document that have been learned independently by the two parallel hierarchical neural networks, which are denoted as $V_{lexical}$ and $V_{syntactic}$ respectively, are concatenated into the final vector representation.

$$V_k = [V_{lexical}; V_{syntactic}]$$

Classification

The learned vector representation of documents are fed into a softmax classifier to compute the probability distribution of class labels. Suppose V_k is the final vector representation of document k output from the fusion layer. The prediction $\tilde{y_k}$ is the output of softmax layer and is computed as:

$$\tilde{y_k} = softmax(W_c V_k + b_c),$$

where W_c and b_c are the learnable weight and learnable bias, respectively; and $\tilde{y_i}$ is a C dimensional vector, where C is the number of classes. We use cross-entropy loss to measure the discrepancy of predictions and the true labels y_k. The model parameters are optimized to minimize the cross-entropy loss over all the documents in the training corpus. Hence, the regularized loss function over X documents denoted by $J(\theta)$ is:

$$J(\theta) = -\frac{1}{X}\sum_{i=1}^{X}\sum_{k=1}^{C} y_{ik} log\tilde{y}_{ik} + \lambda||\theta||$$

Experimental Studies

First, we provide ablation studies to report the contribution of the three stylistic levels (lexical, syntactic, and structural) in the final results. Then we show the performance of our proposed method (Style-HAN) on several benchmark datasets in comparison with the existing baselines in the literature.

Datasets

We evaluate the proposed approach on several benchmark datasets:

- **CCAT10 , CCAT50:** Newswire stories from Reuters Corpus Volume 1 (RCV1) written by 10 and 50 authors, respectively [88].
- **BLOGS10, BLOGS50:** Posts written by 10 and 50 top bloggers respectively, originated from data set of 681,288 blog posts by 19,320 bloggers for blogger.com [79].

Some statistics on the sentence length and document length for each dataset are presented in Table 5.1.

Table 5.1: Dataset Statistics ($|A|$: the number of authors, s: the average number of documents per author, w : the average number of words per doc, n: the average number words per sentence, m: the average number of sentences per document)

Param	CCAT10	CCAT50	BLOGS10	BLOGS50
$\|A\|$	10	50	10	50
S	100	100	874	682
W	580	584	380	331
N	27	26	18	17
M	21	21	24	21

Baselines

We compare our method with various baseline approaches which represent the current state of the art in the authorship attribution problem. In what follows we describe each of the baselines. The results reported in this paper are obtained from the corresponding papers.

- **SVM-affix-punctuation 3-grams** [76]: A traditional machine learning model that feeds affix and punctuation n-grams as syntactic features to an SVM classifier.
- **CNN-char** [74]: A hybrid Convolutional Neural Network that leverages character and word sequences. In this model, characters are represented as one-hot vectors (discrete representations).
- **Continuous N-gram representation** [77]: A feed forward neural model which encodes each character n-gram as a continuous vector representation.
- **N-gram CNN** [85]: A Convolutional Neural Network that takes a sequence of character n-grams as input. Despite the previous neural network models that use either a sequence of words or a sequence of characters as inputs, this model captures local interactions at the character level.

- **Syntax-CNN** [100]: A Convolutional Neural Network that encodes syntax tree of sentences combined with character-level n-grams.

Hyperparameter Tuning

The model hyperparameters include the number of sentences per document(M) and the number of words per sentence(N), with their best values obtained during the tuning phase. Table 5.2 shows the corresponding values for each dataset. The networks are trained using mini-batches with size of 32. We use Nadam optimizer [91] to optimize the cross entropy loss over 50 epochs of training. We use 100 dimensional pre-trained Glove embeddings [63] for the lexical layer and 100 dimensional randomly initialized embeddings for the syntactic layer. In order to reduce the effect of the out-of-vocabulary problem in the lexical layer, we retain only the 50,000 most frequent words. In word-level encoders (Lexical-CNN and Syntactic-CNN), we use 3 parallel convolutional layers with the receptive field sizes (r) of 3, 4, and 5 ($Z = \{3, 4, 5\}$). The total number of filters (K) in each convolution layer is set to 300. In sentence-level encoders, the size of hidden layers in LSTM is set to 128 and 256 for CCAT and BLOGS datasets respectively.

Table 5.2: The model hyperprameters for each dataset

Dataset	**M**	**N**
CCAT10	30	40
CCAT50	30	40
BLOGS10	20	40
BLOGS50	20	40

We partitioned the datasets into train/validation/test in all of our experiments in the same way as they are used in the literature in order to have a fair comparison with the state-of-art methods [74, 100]. All the performance metrics are the mean of accuracy (on the test set) calculated over

10 runs with a 0.9/0.1 train/validation split. For the BLOGS10, BLOGS50 datasets we followed the literature and randomly split off 10% of the dataset as a stratified test set. For CCAT10 and CCAT50, the test set and train set are separated in the original dataset and each author has 50 training samples and 50 test samples. Table 5.3 shows the number of train and test samples for each dataset.

Table 5.3: The number of train and test samples for each dataset

	CCAT10	**CCAT50**	**BLOGS10**	**BLOGS50**
Train	500	2,500	7,866	30,676
Test	500	2,500	875	3,409

Performance Results

Syntactic Representation

First, we compare our proposed syntax encoding method (POS encoding) to the prior method syntax tree (ST) encoding [100]. In ST encoding, the syntax parse tree of sentences are utilized to encode the syntactic information of sentences. Each word in the sentence is embedded through the corresponding path in the syntax tree. In this approach, the hierarchical structure of sentences are explicitly given as an input to the model. However, we argue if such an explicit annotation is necessary for author attribution. In our proposed POS encoding model, each word is embedded by only its part of speech tag and the neural model itself implicitly learns the dependencies between the parts of speech in the sentences. Furthermore, utilizing only POS tags of words makes the model computationally less expensive when compared to utilizing syntax parse tree structure.

Table 5.4 reports the accuracy of different syntactic representations for all the benchmark datasets. In ST encoding, the authors uses a CNN-based neural model; hence, we employ the the identi-

Table 5.4: The accuracy of different syntactic representations

Model	CCAT10	CCAT50	BLOGS10	BLOGS50
ST-CNN	22.8	10.08	48.64	42.91
POS-CNN	61.40	40.98	68.26	54.85
POS-HAN	**63.14**	**41.30**	**69.32**	**57.76**

cal network architecture proposed in the study in order to have a fair comparison of two different syntactic representations. The results for ST encoding are reported from the corresponding paper. The experimental results demonstrate that our proposed syntactic representation (POS-CNN) outperforms the previously proposed method (ST-CNN) by a large margin in all the benchmark datasets (38.6% in CCAT10, 30.80% in CCAT50, 19.62% in BLOGS10, 11.94% in BLOGS50). This improvement in performance can be due two factors. First, the model complexity in POS encoding has been remarkably decreased which makes it more capable of generalization. Second, utilizing syntax tree imposes the positional factor of syntactic units in the sentences. Whereas the authorship attribution task is interested to capture the frequent syntactic patterns, regardless of their position in the sentences.

Hierarchical Neural Model

We have employed hierarchical attention network (HAN) in order to capture the structural information of documents. In order to understand the contribution of our network architecture to the performance, we compare our network architecture (POS-HAN) to the previously proposed CNN-based model (POS-CNN) when the syntactic representations are kept identical. According to Table 5.4, POS-HAN outperforms POS-CNN model consistently across all the benchmark datasets (1.74% in CCAT10, 0.32% in CCAT50, 1.06% in BLOGS10, 2.91% in BLOGS50). This observation indicates that hierarchical neural models which capture the hierarchical structure of documents are a better choice for the authorship attribution task. This confirms our argument that structural information of the document is important to reveal the authorial writing style.

Lexical and Syntactic Model

In order to understand the contribution of lexical and syntactic models to the final predictions, we performed an ablation study. The results are reported in table 7.3. In Syntactic-HAN, only syntactic representation of documents ($V_{syntactic}$) is fed into the softmax layer to compute the final predictions. Similarly, in Lexical-HAN, only lexical representation of documents ($V_{lexical}$) is fed into the softmax classifier. The final stylometry model, Style-HAN, fuses both representations and computes the class labels using a softmax classifier (Section 5). According to the table, lexical model consistently outperforms the syntactic-model across all the benchmark datasets. Moreover, combining the two representations further improves the performance results.

Table 5.5: The accuracy of syntactic (Syntactic-HAN), lexical (Lexical-HAN),and combined (Style-HAN) models

Model	**CCAT10**	**CCAT50**	**BLOGS10**	**BLOGS50**
Syntactic-HAN	63.14	41.30	69.32	57.76
Lexical-HAN	86.04	79.50	70.81	59.77
Style-HAN	**90.58**	**82.35**	**72.83**	**61.19**

Figure 5.4 illustrates the training loss of the syntax, lexical, and style encoding over 50 epochs of training for all the datasets. As we observe, the lexical model maintains lower loss in the earlier epochs and converges faster when compared to the syntactic model. However, combining them into the style model reduces the loss and improves the performance.

Based on the observation from figure 5.4 and table 7.3, we realize that even if Syntactic-HAN achieves comparable performance results, combining it with Lexical-HAN slightly improves the overall performance (Style-HAN). This can be due to the fact that lexical-based recurrent neural networks alone are able to encode significant amount of syntax, even in the absence of explicit syntactic annotations [17]. However, explicit syntactic annotation further improves the performance results when it's compared to the lexical-based model. As shown in Table 7.3, the performance improvement in terms of accuracy is consistent across all the benchmark datasets (4.54% in CCAT10, 2.85% in CCAT50, 2.02% in BLOGS10, 1.42% in BLOGS50).

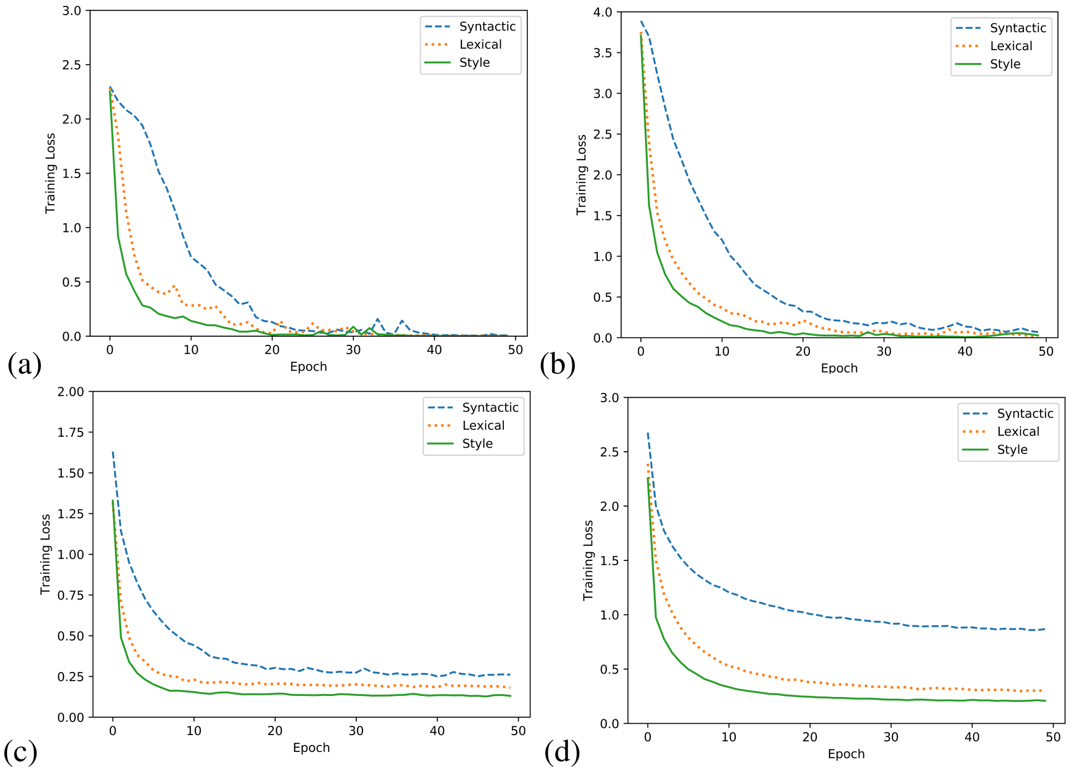


Figure 5.4: Cross Entropy loss over 50 epochs of training for syntactic, lexical, and style models for (a) CCAT10, (b) CCAT50 , (c) BLOGS10 , and (d) BLOGS50 datasets

Training Syntactic and Lexical Networks

We examine two different approaches (combined and parallel) for fusing lexical and syntactic encoding into the final style network. In the combined approach, we concatenate the syntactic and lexical embeddings and construct a unified representation for each word which contains both lexical and syntactic information. Subsequently, this representation is fed to a hierarchical attention network to learn the final document representation. In the parallel approach, the lexical and syntactic embeddings are fed into two identical hierarchical neural networks and the syntactic and lexical representations of documents, which are learned independently and in parallel, are concatenated into a final document representation. Figure 7.1 illustrates these two approaches. Table 5.6 reports the accuracy of the combined and the parallel fusion approaches. According to these results, training two parallel networks for lexical and syntax encoding achieves higher accuracy when compared to training the same network with combined embeddings. This observation can be due to the fact that syntactic and lexical models contain almost complementary information, which are language structure and semantics, respectively. Hence, training them independently delivers better results.

Table 5.6: The accuracy of different fusion approaches

Dataset	**Combined**	**Parallel**
CCAT10	88.36	90.58
CCAT50	81.21	82.35
BLOG10	67.38	72.83
BLOG50	58.81	61.19

Style Encoding

We compare our proposed style-aware neural model (Style-HAN) with the other stylometric models in the literature. Table 5.7 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. The best performance result for each dataset is highlighted in bold. It shows that Style-HAN outperforms the baselines by 2.38%, 1.35%, 8.73%, and

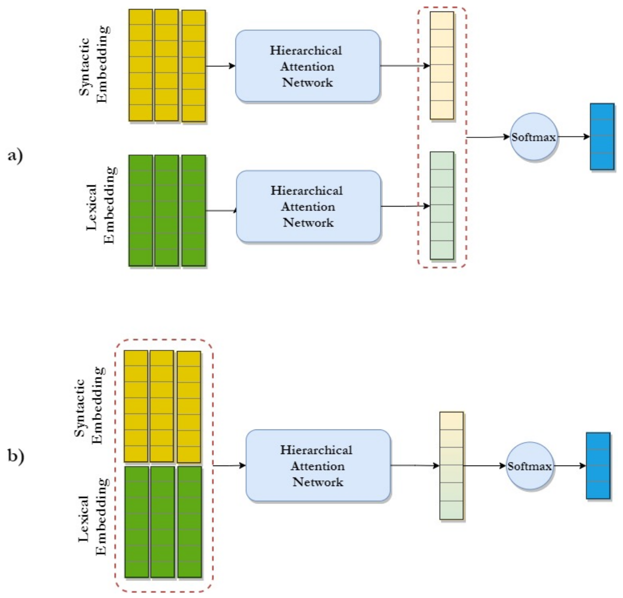

a)
Syntactic Embedding
Lexical Embedding
Hierarchical Attention Network
Hierarchical Attention Network
Softmax
b)
Syntactic Embedding
Lexical Embedding
Hierarchical Attention Network
Softmax

Table 5.7: Test accuracy of models for each dataset

Model	CCAT10	CCAT50	BLOGS10	BLOGS50
SVM-affix-punctuation 3-grams	78.8	69.3	#	#
CNN-char	#	#	61.2	49.4
Continuous n-gram	74.8	72.6	61.34	52.82
N-gram CNN	86.8	76.5	63.74	53.09
Syntax-CNN	88.20	81.00	64.10	56.73
Style-HAN	**90.58**	**82.35**	**72.83**	**61.19**

4.46% over the CCAT10, CCAT50, BLOGs10, and BLOGS50 datasets, respectively. This indicates the effectiveness of encoding document information in three stylistic levels including lexical, syntactic and structural.

Sensitivity to Sentence Length

We examine our model's sensitivity to the sentence length (M). We evaluate the performance of the model on different sentence lengths of 10, 20, 30, and 40 words while the sequence length (N) is kept constant. Figure 5.5 shows the performance results of the Style-HAN on the four datasets. It shows that the model achieves the best performance in CCAT10 and CCAT50 when the sentence length is equal to 30; while in BLOGS10 and BLOGS50, the highest performance is observed when the sentence length is equal to 20. Table 5.1 shows the average sentence lengths in all samples in CCAT10, CCAT50, BLOGS10, and BLOGS50 are 27, 26, 18, and 17, respectively. Accordingly, the models perform better when the sentence length is close to the average sentence length in the dataset. This is simply because a shorter sentence length results in information loss and, on the other hand, a longer sentence length leads to capturing misleading features. Both situations result in lower accuracy.

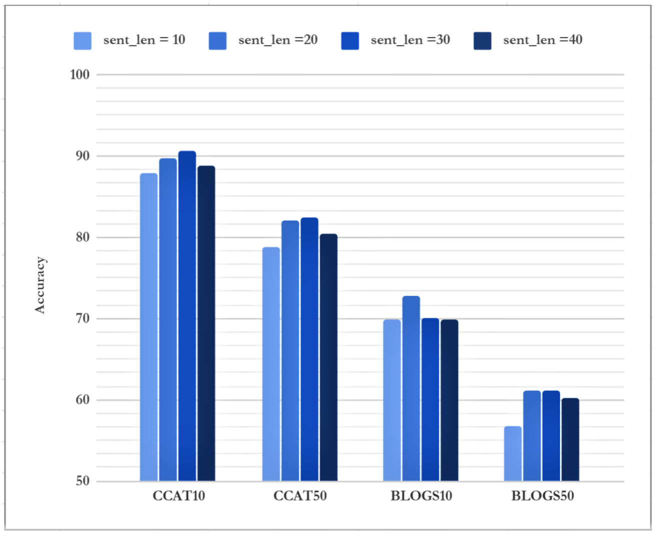


Figure 5.5: The accuracy of Style-HAN model across different sentence length

Sensitivity to Document Length

We examine the model's performance across different numbers of sentences per document (document length). Figure 5.6 illustrates the accuracy of the model when the number of sentences per document is assumed to be 10, 20, 30, and 40, respectively. We observe that increasing the sequence length (the number of sentences in document) generally boosts the performance on all the datasets. This observation confirms the fact that investigation of writing style in short documents is more challenging [57].

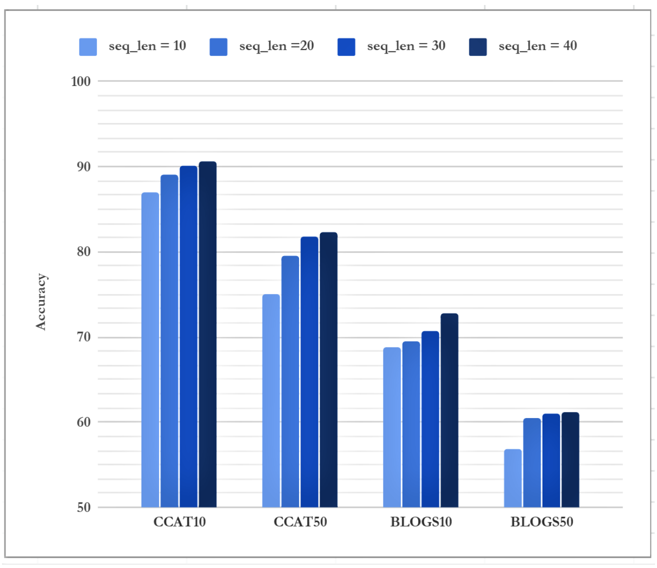


Figure 5.6: The accuracy of Style-HAN model across different document length

Summary

In this chapter, we presented a style-aware neural model which encodes document information from three stylistic levels: lexical, syntactic, and structural in order to better capture the authorial writing style. First, we propose an efficient way to encode the syntactic patterns of sentences using only their corresponding part-of-speech tags. Lexical and syntactic embeddings of words are then used to create two different sentence representations. Subsequently, a hierarchical neural network is employed to capture the structural patterns of sentences in the document, which takes both syntactic and lexical information as input. Finally, these syntactic and lexical representation of documents are concatenated in the fusion step to build the final document representation. Our experimental results on the benchmark datasets in authorship attribution tasks confirm the benefits of encoding document information from all three stylistic levels, and show the performance advantages of our techniques.

CHAPTER 6: TRANSFER LEARNING FOR AUTHORSHIP ATTRIBUTION USING DEEP CONTEXTUALIZED WORD REPRESENTATIONS

[1] We adopt a transfer learning approach using deep contextualized word representations (ELMo) and fine-tune our previously proposed hierarchical neural model. This approach allows us to measure the contribution of lower-level linguistic representations (which mainly carry syntactic information) versus higher-level linguistic representations (which mainly carry semantic information) in the task of authorship attribution. In what follows, we elaborate on our approach.

Word Embeddings

Word embeddings, which are building blocks of any NLP tasks, fall into two categories: *Static* word embeddings which are fixed pre-trained vector representations and *contextualized* word embeddings which are dynamic vector representations and address the polysemous and content-dependent nature of words. In each category, several approaches have been developed to produce word embeddings which mainly differ in how they model the semantics and context of the words. The choice of word embeddings for any particular NLP tasks is still a matter of experimentation and evaluation.

[1]The findings from this research has been previously published:Jafariakinabad, Fereshteh, and Kien A. Hua. "Unifying Lexical, Syntactic, and Structural Representations of Written Language for Authorship Attribution." SN Computer Science 2.6 (2021): 1-14.

Static Embeddings

We use pre-trained Glove [63] and FastText [18] embeddings and represent each sentence as the sequence of its corresponding word embeddings. Glove is a count-based model which relies on the distributional language hypothesis in order to capture the semantics. FastText is a character-based model in which a word is represented as a bag of character n-grams and the final word vector is the sum of these representations. One of the advantages of this representation is the capability of handling out-of-vocabulary words.

Contextualized Embeddings

Even though the static word embeddings capture semantics of words to some extent, they fail to capture polysemy and disregard the context in which the word appears. To address the polysemous and context-dependent nature of words, the contextualized word embeddings are proposed [27, 3, 65]. Embeddings from Language Modeling Objective (ELMo) proposes a deep contextualized word representation in which each representation is a function of the input sentence. The objective function is a 2-layer bidirectional language model (BiLM) and the final representations are a linear combination of all of the internal layers of the BiLM and the weights are learnable for a specific task [65] (Fig 6.1). One of the interesting characteristics of ELMo embedding is that different layers in this language model learn different kinds of representation. The higher-level BiLM mainly capture context-dependent information while the lower-level states mainly capture syntactic information [65]. This scheme allows us to measure the contribution of semantics versus syntactic aspects of the words in our authorship attribution task.

The ELMo model learns the contextualized representation of words from three levels. The first layer is the context-free representation which are the inputs of a bidirectional language model consisting of two bidirectional LSTMs. The final ELMo embedding is the weighted linear combination of these three layers. Let's assume that the hidden state of the model corresponding to the k^{th} word in the layer i is denoted as $h_{k,i}^{LM}$, hence the final contextualized representation is calculated as $ELMo_k^{task} = \gamma^{task} \sum_{i=0}^{2} s_i^{task} h_{k,i}^{LM}$, where s_i^{task} are trainable task-specific weights for each layer and γ^{task} is a scalar parameter to scale the final ELMo vector representations.

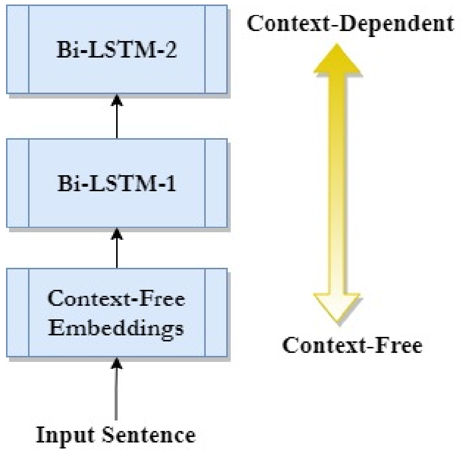


Figure 6.1: The Architecture of ELMo Embedding

Experimental Results

In this section we experiment with different pre-trained word-embeddings. In all of the experiments, we use previously proposed hierarchical neural model presented in Chapter 5. First, in order to compare the effects of lower-level linguistic representations to the higher-level linguistic representations, we use a transfer learning approach and fine-tune our hierarchical model by using pre-trained ELMo embeddings in the lexical level. In ELMo model, the higher-level states mainly capture context-dependent information while the lower-level states mainly capture syntactic information [65]. This scheme allows us to measure the contribution of semantics versus syntactic aspects of the words in our authorship attribution task. During the training phase, task-specific weights (γ^{task}) are learned.

Figure 6.2 illustrates the performance of context-free embeddings (word-level), the first LSTM layer (lstm1_layer), the second LSTM layer(lstm2_layer), the weighted sum of all layers (weighted_layers), and the fixed mean-pooling of layers [2].

According to the results, word-level representation which is a context-free representation of ELMo consistently achieves higher performance compared to the two BiLM layers which are both context-dependent representations. In BiLM layers the first layer achieves better performance across all datasets when compared to the second layer. Hence, it is observed from the results that lower-level linguistic representations of ELMo which mainly capture syntactic information demonstrate a better performance in authorship attribution task when compared to higher-level representations of ELMo which mainly carry semantic information. Finally the weighted sum of all layers or mean-pooling of layers do not improve the performance when compared to individual layers and, unsurprisingly, the weighted sum of all layers shows a better performance compared to the mean-pooling.

Table 6.1: The accuracy of different lexical representations

Model	**CCAT10**	**CCAT50**	**BLOGS10**	**BLOGS50**
ELMo_word_level	77.00	62.44	75.41	66.27
ELMo_lstm1	73.20	59.12	75.24	57.62
ELMo_lstm2	62.20	48.56	73.07	42.10
ELMo_Weighted	70.20	53.40	66.49	65.13
ELMo_mean	55.60	45.76	59.86	36.65
FastText-HAN	81.00	70.44	68.21	58.38
Glove-HAN	86.04	79.50	70.81	59.77

[2] `https://tfhub.dev/google/elmo/3`

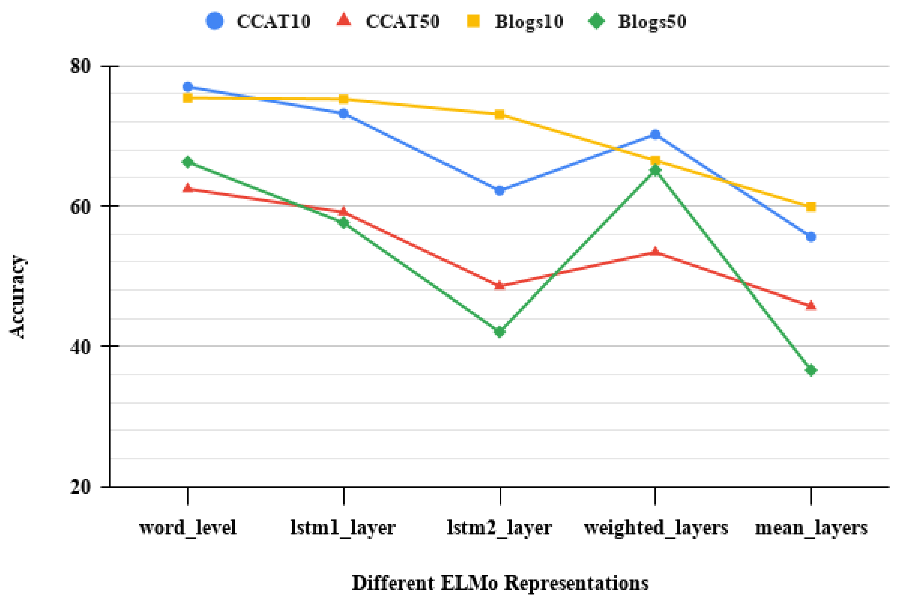


Figure 6.2: The performance of different ELMo representations

We have also experimented with two static word-embeddings, Glove [63] and Fasttext [18]. Table 6.1 reports the performance of different word representations across four datasets. Glove consistently outperforms Fasttext across all the datasets. When comparing Glove against ELMo embeddings, none of them outperforms the other consistently. According to the results, Glove achieves higher accuracy compared to ELMo in CCAT datasets by 9.04% in CCAT10 and 17.06% in CCAT50. However, ELMo word-level embedding performs better in BLOGS datasets by 4.6% in BLOGS10 and 6.5% in BLOGS50 when compared to Glove embeddings. This is due to the fact that ELMo is a dynamic complex neural network and training such a model demands more training samples to achieve a higher performance. Consequently, due to the lower number of training samples in the CCAT datasets, ELMo does not achieve higher performance results.

Summary

In this chapter, we presented a transfer learning approach using deep contextualized word representation in order to measure the impact of different linguistic representation in the task of authorship attribution. According to our experimental results, lower-level linguistic representations which mainly carry syntactic information, demonstrate better performance in authorship attribution task when compared to higher-level linguistic representations, which mainly carry semantic information.

Additionally, we compared the performance of static word embeddings (Glove, FastText) against deep contextualized word embeddings(ELMo). Our experimental results indicates that the performance of ELMo is highly dependent on the size of dataset. This is observation is due to the fact that ELMo is a dynamic complex neural network and training such a model demands more training samples to achieve its potential performance.

CHAPTER 7: A SELF-SUPERVISED REPRESENTATION LEARNING OF SENTENCE STRUCTURE FOR AUTHORSHIP ATTRIBUTION

Proposed Framework : Lexicosynt Network

[1] The Lexicosynt network is a Siamese network [23] which comprises lexical and syntactic sub-networks and a loss function. Figure 7.1 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.

Lexical Sub-network

The lexical sub-network encodes the sequence of words in a sentence (Figure 7.2 (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. [52]. 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.

[1]The findings from this research has been accepted for publication in ACM Transactions on Knowledge Discovery from Data. Jafariakinabad, Fereshteh, and Kien A. Hua. ”A self-supervised representation learning of sentence structure for authorship attribution.” arXiv preprint arXiv:2010.06786 (2020).

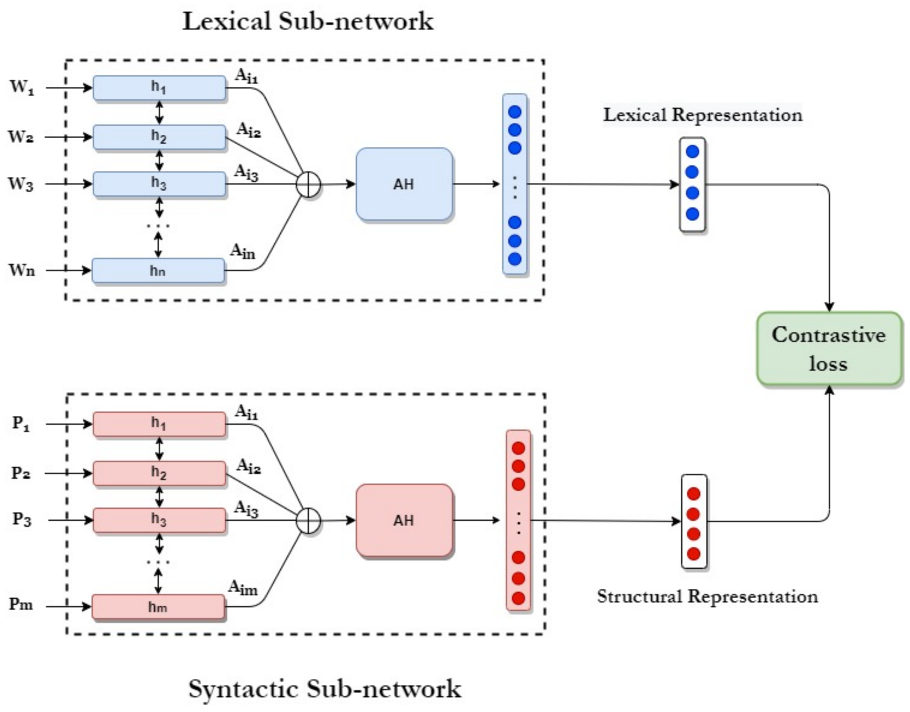


Figure 7.1: The overall architecture of LexicoSynt network

$$S = (W_1, W_2, ..., W_n),$$

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(W_t, \overrightarrow{h_{t-1}}),$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(W_t, \overleftarrow{h_{t+1}}),$$

$$H = (h_1, h_2, ..., h_n),$$

$$A = softmax(W_{s2} tanh(W_{s1} H^T)),$$

$$M = AH.$$

Syntactic Sub-network

The syntactic sub-network aims to encode 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 7.2 (b). To adapt the tree representation to recurrent neural networks, we linearize the syntax parse tree to a sequence of structural labels. Figure 7.2 (c) shows the structural label sequence of Figure 7.2 (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.

Loss Function

Given a sentence, we aim to minimize 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 [23], 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 :

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

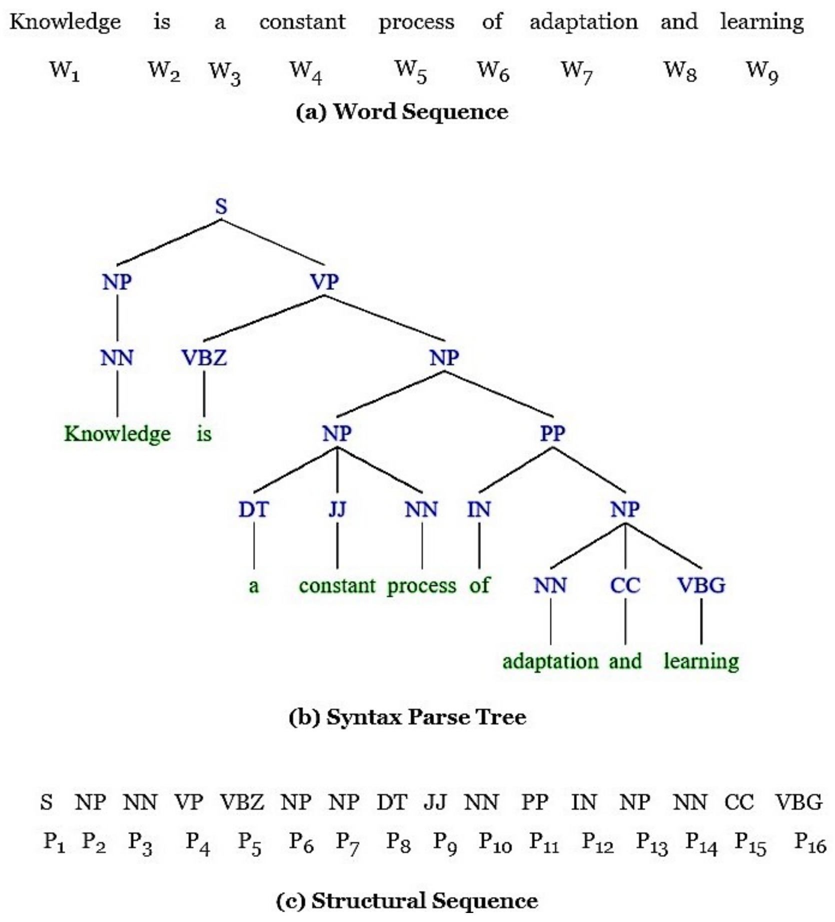

Knowledge is a constant process of adaptation and learning
W1 W2 W3 W4 W5 W6 W7 W8 W9
(a) Word Sequence
S
NP
VP
NN
VBZ
NP
Knowledge
is
NP
PP
DT
JJ
NN
IN
NP
a
constant
process
of
NN
CC
VBG
adaptation
and
learning
(b) Syntax Parse Tree
S NP NN VP VBZ NP NP DT JJ NN PP IN NP NN CC VBG
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
(c) Structural Sequence

Figure 7.2: An example of an input sentence and the corresponding (a) word sequence representation (b) syntax parse tree representation (c) linearized parse tree representation known as structural sequence

$V_{lexical}$ 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.

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 paper reproducible, we have made our framework implementations publicly available [2].

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

Data

Training Data:

The proposed model has been trained on LAMBADA (LAnguage Modeling Broadened to Account for Discourse Aspects) dataset [61] 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 1000 sentences are randomly selected for the development set.

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 [88].
- **BLOGS10, BLOGS50:** Posts written by 10 and 50 top bloggers respectively, originated from data set of 681,288 blog posts by 19,320 bloggers for blogger.com [79].

Training

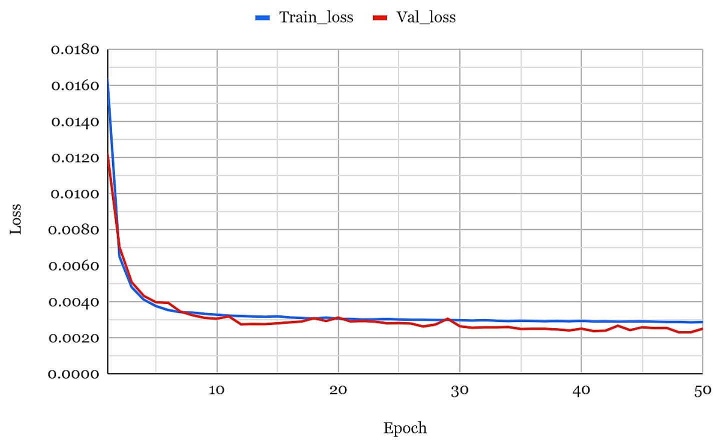
For the input data of the syntactic sub-network, we have generated the parse tree of each sentence in the training set using CoreNLP parser [55]. Each sentence and its corresponding linearized parse tree is fed into the network as a genuine (or positive) pair. To generate the false (negative) pairs, we have paired each sentence in the batch with a randomly selected linearized parse tree in 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 1000 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 initialised using uniform Xavier initialization. Both the 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 7.3 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.

Representation Learning Evaluation: Probing Tasks

We use 10 probing tasks introduced by Conneau et al. [25] 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 [3]. The probing tasks are grouped into three classes including surface in-

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



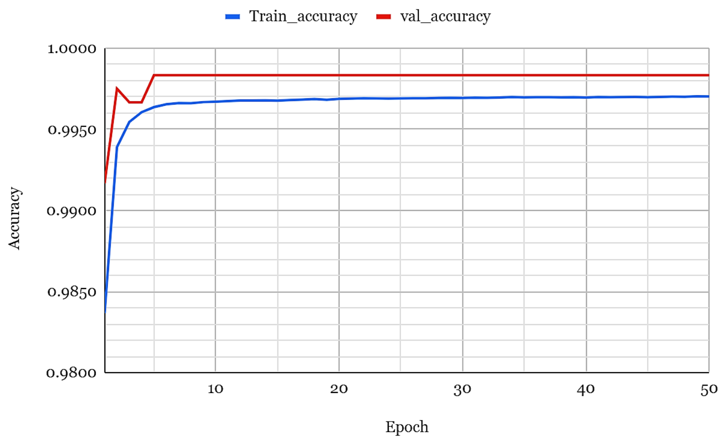


Figure 7.3: The training and validation loss and accuracy over 50 epochs of training

formation, 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.

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.

Figure 7.4 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 7.1). This implies that structural information of words in the sentence improves the recovery of its content.

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.

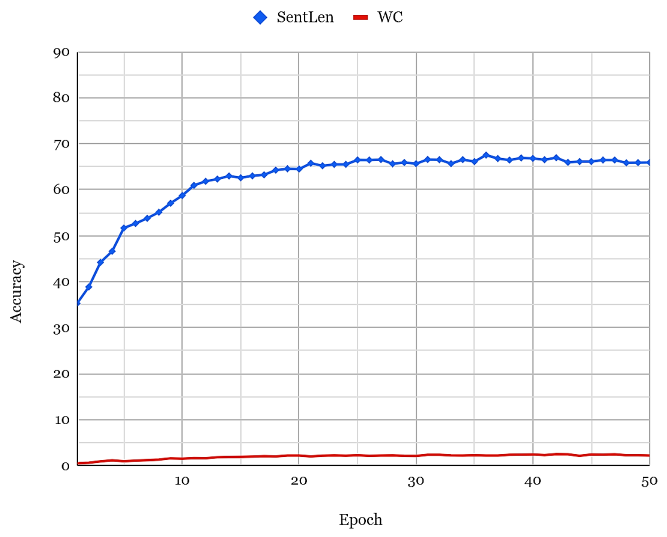


Figure 7.4: The accuracy of learned surface information over 50 epochs of training

Figure 7.5 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.

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.

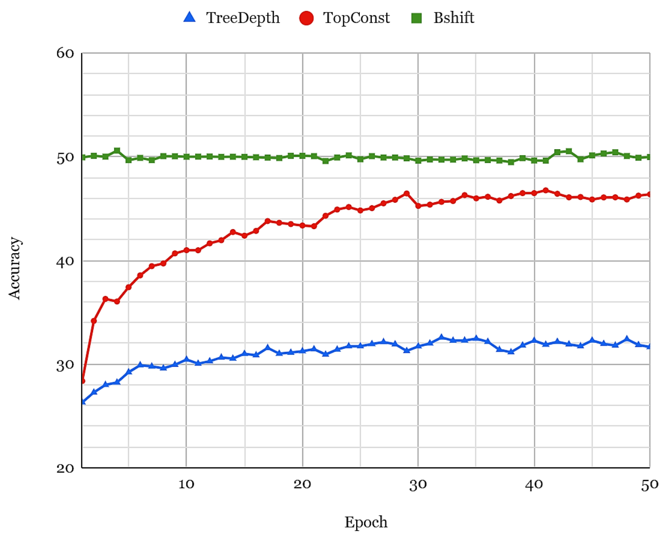


Figure 7.5: The accuracy of syntactic information over 50 epochs of training

- *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 7.6 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 per-

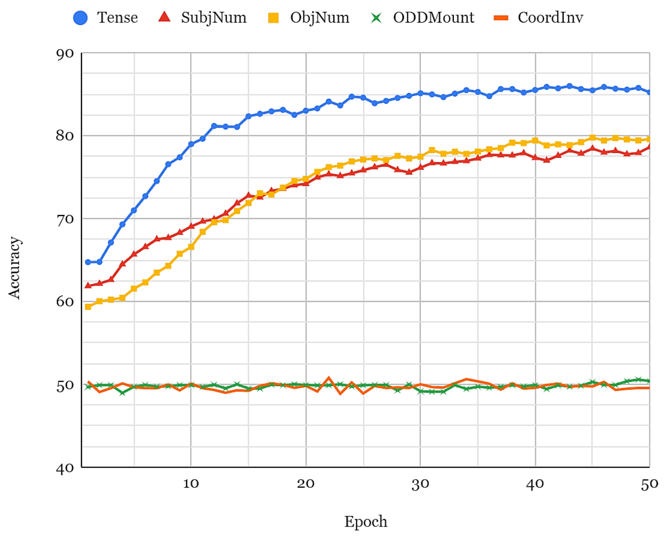


Figure 7.6: The accuracy of semantic information over 50 epochs of training

formance in the tasks where both semantic and structural information is required (e.g. Tense, 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.

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 [63] and FastText [18]. We use BoV representation of sentences for both its simplicity and its capability at capturing sentence level properties [25]. Table 7.1 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 [25] 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 7.1: 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

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 7.2 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 [10]. In SelfATT_AVGPool_seq4040 and SelfATT_MaxPool_seq4040, we incorporate Self-attention mechanism [52] 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

Table 7.2: 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**

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).

Test on Authorship Attribution Datasets

In our previous work [41], 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 7.7. 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 part-of-speech (POS) tag in the sentence, and then embed each POS tag into a low dimensional vector $P_i \in R^{d_p}$ using a trainable lookup table $\theta_P \in R^{|T| \times d_p}$, 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.

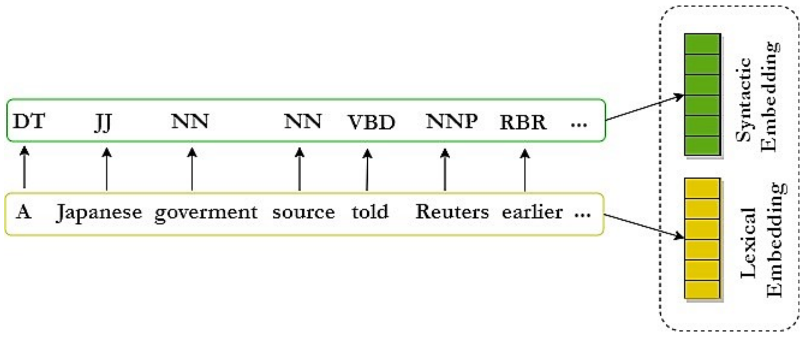

DT JJ NN NN VBD NNP RBR ...
A Japanese goverment source told Reuters earlier ...
Syntactic Embedding
Lexical Embedding

Figure 7.7: Lexical and Syntactic Embedding

We have compared our proposed style-aware neural model (Style-HAN) with the other stylometric models in the literature, including Continuous N-gram representation [77], N-gram CNN [85], and syntax-CNN [100]. Table 7.3 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 [41], only syntactic representation of documents is fed into the softmax layer to compute the final predictions. Similarly, in Lexical-HAN [41], only lexical representation of documents is fed into the softmax classifier. The final stylometry model, Style-HAN, fuses both representations. In order to examine the efficacy of our proposed structural embeddings in this paper 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 7.3, the best performance result for each dataset is highlighted in bold. It shows that

Table 7.3: 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**

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 part-of-speech 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 part-of-speech 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 part-of-speech 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.

Summary

In this paper, 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 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.

CHAPTER 8: CONCLUSION AND FUTURE WORK

This dissertation addressed topic detection and authorship attribution for user-generated textual data. In Chapter 3, we presented a maximal sequence mining approach for the problem of topic detection from microblog streams. Experiments on Twitter datasets demonstrate that the proposed model achieves high performance in comparison with the state-of-the-art methods. Afterward, we mainly focus on the task of authorship attribution. Generally, the problem of determining the style of a document is orthogonal to the problem of determining its topic, since the document features which capture the style are mainly independent of the document's topic.

Writing style in written language is a combination of consistent decisions associated with a specific author at different levels of language production: lexical, syntactic, and structural. The recent works in neural network-based style analysis mainly focus on lexical-based neural models and lack the multi-level modeling of writing style. In Chapter 4, we presented our proposed neural models to encode syntactic and structural patterns of sentences for the task of authorship attribution. In Chapter 5, we presented a unified neural model that encodes documents in three stylistic levels. Our experimental results, based on four authorship attribution benchmark datasets, reveal the benefits of encoding document information from all three stylistic levels when compared to the baseline methods in the literature. In Chapter 6, we elaborated on how to utilize deep contextualized word representations (ELMo) and adopt a transfer learning approach to fine-tune our previously proposed model. The proposed approach is utilized to measure the performance of lower-level linguistic representations versus higher-level linguistic representations of ELMo in the task of authorship attribution. Finally, in Chapter 7 we presented a self-supervised representation learning framework to learn the structural representation of sentences. Such pre-trained syntactic embeddings can be used in conjunction with lexical embeddings in the domain of authorship attribution. It is worth mentioning that pre-trained syntactic embeddings eliminate the necessity of syntactic parsing of the input sentences in our previously proposed model; hence, the model will be computationally more efficient.

In this dissertation, we have shown the benefits of multi-level modeling of writing style in neural network-based models in the authorship attribution task which was previously lacking in the literature. Potential future extensions of this work can explore different model architectures, extracting and representing additional stylistic features, and adapting the proposed model to the other areas of computational stylometry, including authorship profiling and similarity detection tasks. Moreover, pre-training of both lexical and syntactic embeddings on the target domain is desirable to further improve the performance of the proposed models in the authorship attribution task.

LIST OF REFERENCES

[1] H. V. Agun, S. Yilmazel, and O. Yilmazel. Effects of language processing in turkish authorship attribution. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1876–1881, 2017. doi: 10.1109/BigData.2017.8258132.

[2] L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. *IEEE Transactions on Multimedia*, 15(6):1268–1282, 2013.

[3] A. Akbik, D. Blythe, and R. Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[4] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. 1998.

[5] B. Alsulami, E. Dauber, R. Harang, S. Mancoridis, and R. Greenstadt. Source code authorship attribution using long short-term memory based networks. In *European Symposium on Research in Computer Security*, pages 65–82. Springer, 2017.

[6] K. Apoorva and S. Sangeetha. Deep neural network and model-based clustering technique for forensic electronic mail author attribution. *SN Applied Sciences*, 3(3):1–12, 2021.

[7] K. Apoorva and S. Sangeetha. Forensic analysis of e-mail for authorship attribution: Research perspective. In *Proceeding of First Doctoral Symposium on Natural Computing Research: DSNCR 2020*, volume 169, page 281. Springer Nature, 2021.

[8] S. Argamon-Engelson, M. Koppel, and G. Avneri. Style-based text categorization: What

newspaper am i reading. In *Proc. of the AAAI Workshop on Text Categorization*, pages 1–4, 1998.

[9] D. Bagnall. Authorship clustering using multi-headed recurrent neural networks. *arXiv preprint arXiv:1608.04485*, 2016.

[10] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[11] R. Banga and P. Mehndiratta. Authorship attribution for textual data on online social networks. In *2017 Tenth International Conference on Contemporary Computing (IC3)*, pages 1–7, 2017. doi: 10.1109/IC3.2017.8284311.

[12] Y. Bao, H. Zhou, S. Huang, L. Li, L. Mou, O. Vechtomova, X. Dai, and J. Chen. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6008–6019, 2019.

[13] S. Barbon, R. A. Igawa, and B. B. Zarpelao. Authorship verification applied to detection of compromised accounts on online social networks. *Multimedia Tools and Applications*, 76 (3):3213–3233, 2017.

[14] H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: Real-world event identification on twitter. *ICWSM*, 11:438–441, 2011.

[15] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[17] T. Blevins, O. Levy, and L. Zettlemoyer. Deep rnns encode soft hierarchical syntax. *arXiv preprint arXiv:1805.04218*, 2018.

[18] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[19] C. Castellano, F. Cecconi, V. Loreto, D. Parisi, and F. Radicchi. Self-contained algorithms to detect communities in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):311–319, 2004.

[20] E. Castillo, D. Vilarino, O. Cervantes, and D. Pinto. Author attribution using a graph based representation. In *2015 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 135–142. IEEE, 2015.

[21] M. Cataldi, L. Di Caro, and C. Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, page 4. ACM, 2010.

[22] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[23] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[24] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, 2017.

[25] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram

into a single \&# vector: Probing sentence embeddings for linguistic properties. In *ACL 2018-56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2126–2136. Association for Computational Linguistics, 2018.

[26] E. Dauber, A. Caliskan, R. Harang, and R. Greenstadt. Poster: Git blame who?: Stylistic authorship attribution of small, incomplete source code fragments. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 356–357, 2018.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[28] E. Ferracane, S. Wang, and R. Mooney. 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, 2017.

[29] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian Conference on Artificial Intelligence*, pages 83–94. Springer, 2014.

[30] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st international conference on Very large data bases*, pages 181–192. VLDB Endowment, 2005.

[31] C. Gallagher and Y. Li. Text categorization for authorship attribution in english poetry. In *Science and Information Conference*, pages 249–261. Springer, 2018.

[32] Z. Ge, Y. Sun, and M. J. Smith. Authorship attribution using a neural network language model. In *AAAI*, pages 4212–4213, 2016.

[33] J. Guo, P. Zhang, L. Guo, et al. Mining hot topics from twitter streams. *Procedia Computer Science*, 9:2008–2011, 2012.

[34] M. Hajja, A. Yahya, and A. Yahya. Authorship attribution of arabic articles. In *International Conference on Arabic Language Processing*, pages 194–208. Springer, 2019.

[35] S. Havrylov, G. Kruszewski, and A. Joulin. 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)*, pages 1118–1128, 2019.

[36] M. Heidari and J. H. Jones. Using bert to extract topic-independent sentiment features for social media bot detection. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0542–0547, 2020. doi: 10.1109/UEMCON51285.2020.9298158.

[37] M. Heidari, J. H. Jones, and O. Uzuner. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 480–487, 2020. doi: 10.1109/ICDMW51313.2020.00071.

[38] J. Hewitt and C. D. Manning. 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)*, pages 4129–4138, 2019.

[39] J. Hitschler, E. van den Berg, and I. Rehbein. Authorship attribution with convolutional neural networks and pos-eliding. In *Proceedings of the Workshop on Stylistic Variation*, pages 53–58, 2017.

[40] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[41] F. Jafariakinabad and K. A. Hua. Style-aware neural model with application in authorship attribution. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 325–328. IEEE, 2019.

[42] F. Jafariakinabad, S. Tarnpradab, and K. A. Hua. Syntactic neural model for authorship attribution. In *The Thirty-Third International Flairs Conference*, 2020.

[43] P. Juola, J. Milička, and P. Zemánek. Authorship and time attribution of arabic texts using jgaap. In *Intelligent Natural Language Processing: Trends and Applications*, pages 325–349. Springer, 2018.

[44] J. Kabala. Computational authorship attribution in medieval latin corpora: the case of the monk of lido (ca. 1101–08) and gallus anonymous (ca. 1113–17). *Language Resources and Evaluation*, 54(1):25–56, 2020.

[45] H. D. Kim, D. H. Park, Y. Lu, and C. Zhai. Enriching text representation with frequent pattern mining for probabilistic topic modeling. *Proceedings of the American Society for Information Science and Technology*, 49(1):1–10, 2012.

[46] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[47] M. Koppel, J. Schler, and S. Argamon. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26, 2009.

[48] M. Krause. A behavioral biometrics based authentication method for mooc's that is robust against imitation attempts. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 201–202. ACM, 2014.

[49] T. Kreutz and W. Daelemans. Exploring classifier combinations for language variety identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 191–198, 2018.

[50] A. Kuncoro, C. Dyer, J. Hale, D. Yogatama, S. Clark, and P. Blunsom. 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)*, pages 1426–1436, 2018.

[51] J. Li, D. Xiong, Z. Tu, M. Zhu, M. Zhang, and G. Zhou. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, 2017.

[52] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[53] T. Litvinova, O. Litvinlova, O. Zagorovskaya, P. Seredin, A. Sboev, and O. Romanchenko. "ruspersonality": A russian corpus for authorship profiling and deception detection. In *2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT)*, pages 1–7, 2016. doi: 10.1109/FRUCT.2016.7584767.

[54] R. Liu, J. Hu, W. Wei, Z. Yang, and E. Nyberg. Structural embedding of syntactic trees for machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 815–824, 2017.

[55] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The

stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[57] T. Neal, K. Sundararajan, A. Fatima, Y. Yan, Y. Xiang, and D. Woodard. Surveying stylometry techniques and applications. *ACM Computing Surveys (CSUR)*, 50(6):86, 2017.

[58] M. E. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[59] M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin*, 29(5):665–675, 2003.

[60] P. Panicheva and T. Litvinova. Authorship attribution in russian in real-world forensics scenario. In *International Conference on Statistical Language and Speech Processing*, pages 299–310. Springer, 2019.

[61] D. Paperno, G. Kruszewski, A. Lazaridou, N.-Q. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. 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)*, pages 1525–1534, 2016.

[62] J. W. Pennebaker and L. A. King. Linguistic styles: Language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296, 1999.

[63] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation.

In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[64] C. S. Perone, R. Silveira, and T. S. Paula. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*, 2018.

[65] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[66] G. Petkos, S. Papadopoulos, L. Aiello, R. Skraba, and Y. Kompatsiaris. A soft frequent pattern mining approach for textual topic detection. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, page 25. ACM, 2014.

[67] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–189. Association for Computational Linguistics, 2010.

[68] S. Phuvipadawat and T. Murata. Breaking news detection and tracking in twitter. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 3, pages 120–123. IEEE, 2010.

[69] S. R. Pillay and T. Solorio. Authorship attribution of web forum posts. In *2010 eCrime Researchers Summit*, pages 1–7. IEEE, 2010.

[70] J.-P. Posadas-Durán, I. Markov, H. Gómez-Adorno, G. Sidorov, I. Batyrshin, A. Gelbukh, and O. Pichardo-Lagunas. Syntactic n-grams as features for the author profiling task. *Working Notes Papers of the CLEF*, 2015.

[71] D. Quercia, H. Askham, and J. Crowcroft. Tweetlda: supervised topic classification and

link prediction in twitter. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 247–250. ACM, 2012.

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

[73] A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, and E. Stamatatos. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33, 2017. doi: 10.1109/TIFS.2016.2603960.

[74] S. Ruder, P. Ghaffari, and J. G. Breslin. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*, 2016.

[75] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

[76] U. Sapkota, S. Bethard, M. Montes, and T. Solorio. 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, 2015.

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

[78] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *Icwsm*, 2009.

[79] J. Schler, M. Koppel, S. Argamon, and J. W. Pennebaker. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 6, pages 199–205, 2006.

[80] R. Schwartz, M. Sap, I. Konstas, L. Zilles, Y. Choi, and N. A. Smith. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. *arXiv preprint arXiv:1702.01841*, 2017.

[81] S. Segarra, M. Eisen, and A. Ribeiro. Authorship attribution through function word adjacency networks. *IEEE Transactions on Signal Processing*, 63(20):5464–5478, 2015.

[82] Y. Seroussi, I. Zukerman, and F. Bohnert. Authorship attribution with latent dirichlet allocation. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 181–189. Association for Computational Linguistics, 2011.

[83] Shaokang Wang and Baoping Yan. Authorship attribution for chinese text based on sentence rhythm features. In *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, pages 61–64, 2010. doi: 10.1109/YCICT.2010.5713152.

[84] Y. Shen, Z. Lin, C.-W. Huang, and A. Courville. Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*, 2017.

[85] P. Shrestha, S. Sierra, F. Gonzalez, M. Montes, P. Rosso, and T. Solorio. 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, 2017.

[86] J. Soler and L. Wanner. 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*, volume 2, pages 681–687, 2017.

[87] K. Song, L. Zhao, and F. Liu. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1717–1729, 2018.

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

[89] E. Stamatatos and M. Koppel. Plagiarism and authorship analysis: introduction to the special issue. *Language Resources and Evaluation*, 45(1):1–4, 2011.

[90] K. Sundararajan and D. Woodard. What represents" style" in authorship attribution? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2814–2822, 2018.

[91] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[92] P. Szwed. Authorship attribution for polish texts based on part of speech tagging. In *International Conference: Beyond Databases, Architectures and Structures*, pages 316–328. Springer, 2017.

[93] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. Van Durme, S. R. Bowman, D. Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

[94] K. Tran, A. Bisazza, and C. Monz. The importance of being recurrent for modeling hierarchical structure. *arXiv preprint arXiv:1803.03585*, 2018.

[95] P. Varela, E. Justino, and L. S. Oliveira. Selecting syntactic attributes for authorship attribution. In *The 2011 International Joint Conference on Neural Networks*, pages 167–172. IEEE, 2011.

[96] J. Weng and B.-S. Lee. Event detection in twitter. *ICWSM*, 11:401–408, 2011.

[97] S.-T. Wu. Knowledge discovery using pattern taxonomy model in text mining. 2007.

[98] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[99] M. Zhang, Z. Li, G. Fu, and M. Zhang. Syntax-enhanced neural machine translation with syntax-aware word representations. In *Proceedings of NAACL-HLT*, pages 1151–1161, 2019.

[100] R. Zhang, Z. Hu, H. Guo, and Y. Mao. Syntax encoding with application in authorship attribution. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2742–2753, 2018.