The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Syntactic Neural Model for Authorship Attribution

Fereshteh Jafariakinabad, Sansiri Tarnpradab, Kien A. Hua
University of Central Florida
Computer Science Department
fereshteh.jafari@knights.ucf.edu

Abstract

Writing style is a combination of consistent decisions at different levels of language production including lexical, syntactic, and structural associated to a specific author (or author groups). While lexical-based models have been widely explored in style-based text classification, relying on content makes the model less scalable when dealing with heterogeneous data comprised of various topics. On the other hand, syntactic models which are content-independent, are more robust against topic variance. In this paper, we introduce a syntactic recurrent neural network to encode the syntactic patterns of a document in a hierarchical structure. The model first learns the syntactic representation of sentences from the sequence of part-of-speech tags. Subsequently, the syntactic representations of sentences are aggregated into document representation using recurrent neural networks. Our experimental results on PAN 2012 dataset for authorship attribution task shows that syntactic neural network outperforms the lexical model with the identical architecture by approximately 14% in terms of accuracy.

Introduction

Individuals express their thoughts in different ways due to many factors including the conventions of language, educational background, 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. Early work on computational stylometry was introduced in the 1960s by Mosteller and Wallace on federalist papers (Mosteller and Wallace 1964). The unprecedented availability of digital data in recent years along with the advancements in machine learning techniques has led to an increase in scholarly attention to the field of Computational stylometry (Koppel, Schler, and Argamon 2009; Neal et al. 2017).

Stylistic features are generally *content-independent* which means that they are mainly 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 sentences in a document. Structural features reveal information about how an author organizes the structure of a document.

One of the basic problems which is rarely addressed in the literature is the interaction of style and content. While content words can be predictive features of authorial writing style due to the fact that they carry information about author's lexical choice, excluding content words as features is a fundamental step for avoiding topic detection rather than style detection (Daelemans 2013). Syntactic and structural features are content-independent which makes them robust against divergence of topics. The frequency of function words, punctuation, and part-of-speech 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. These count-based features are mostly used as the inputs to the conventional machine learning techniques.

The adopted approaches in deep neural networks for style-based text classification mainly focus on lexical features despite the fact that lexical-based language models have very limited scalability when dealing with datasets containing diverse topics and genre (Sundararajan and Woodard 2018). While previously proposed deep neural network approaches focus on lexical level, we introduce 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.

Related Work

Syntax for Style Detection

Style-based text classification was primarily proposed by Argamon-Engelson et al. (Argamon-Engelson, Koppel, and Avneri 1998). The authors used basic stylistic features (the frequency of function words and POS trigrams) to classify news documents based on the corresponding publisher (newspaper or magazine) as well as text genre (editorial or news item). Syntactic n-grams are shown to achieve promis-

ing results in different stylometric tasks (Krause 2014; Posadas-Durán et al. 2015; Sundararajan and Woodard 2018; Kreutz and Daelemans 2018). In particular, Raghavan 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 (Raghavan, Kovashka, and Mooney 2010).

Neural Network in Stylometry

With the recent advances in deep learning, there exists a large body of work in the literature which employs deep neural networks for stylometry and authorship attribution. For instance, Ge et al. used a feed-forward neural network language model on an authorship attribution task. The output achieves promising results compared to the n-gram baseline (Ge, Sun, and Smith 2016). Bagnall et al. have employed a recurrent neural network with a shared recurrent state which outperforms other proposed methods in PAN 2015 task (Bagnall 2016).

Shrestha et al. applied CNN based on character n-gram to identify the authors of tweets. Given that each tweet is short in nature, their approach shows that a sequence of character n-grams as an input to CNN allows the architecture to capture the character-level interactions, which afterwards is aggregated to learn higher-level patterns for modeling the style (Shrestha et al. 2017). Hitchler et al. propose a CNN based on pretrained 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 (Hitschler, van den Berg, and Rehbein 2017).

The Proposed Model: Syntactic Neural Model

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 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 attention mechanism to highlight the sentences which contribute more to the prediction of labels. Afterwards we use a softmax classifier to compute the probability distribution over class labels. The overall architecture of the network is shown in figure 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 $P_i \in \mathbb{R}^{d_p}$ using a trainable lookup table $\theta_P \in \mathbb{R}^{|T| \times d_p}$, where T is the set of all possible POS tags in the language. We use NLTK part-of-speech tagger (Bird, Klein, and Loper 2009) for the tagging purpose and use the set of 47 POS tags[1] 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 50K words. On the other hand, 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.

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 $W \in \mathbb{R}^{rd_p}$ 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 $C_i^r \in \mathbb{R}^{N-r+1}$. 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 $h_i \in \mathbb{R}^K$ 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 $h_t^p \in \mathbb{R}^{2d_l}$ 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, $h_i^s \in \mathbb{R}^{2d_l}$ 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.

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

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

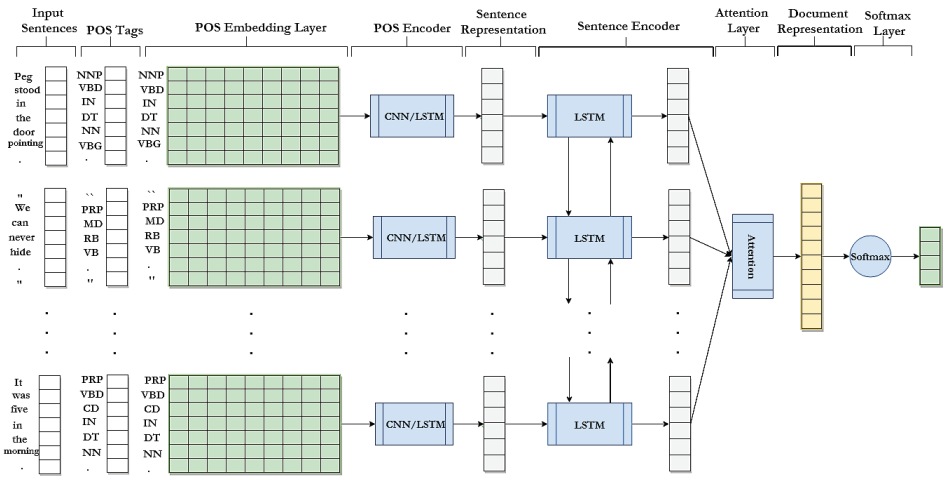

Figure 1: The Overall Architecture of Syntactic Neural Model

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

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

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

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.

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

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

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

Needless to say, 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 and, V is the vector representation of 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 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[2]. 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 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

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

	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

Table 1: Corpus Statistics.

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.

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 the identical architecture. We use 300-dimensional pretrained Glove embeddings (Pennington, Socher, and Manning 2014) for the embedding layer in the network.

Hyperparameter Tuning

In this part, 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 (Sutskever et al. 2013) to optimize the cross-entropy loss over 30 epochs of training.

CNN for POS encoding Figure 2 illustrates the performance of 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 higher 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 improve the performance. Therefore, in the final model, we use one layer of multiple convolutional filters with the receptive field sizes of 3 and 5.

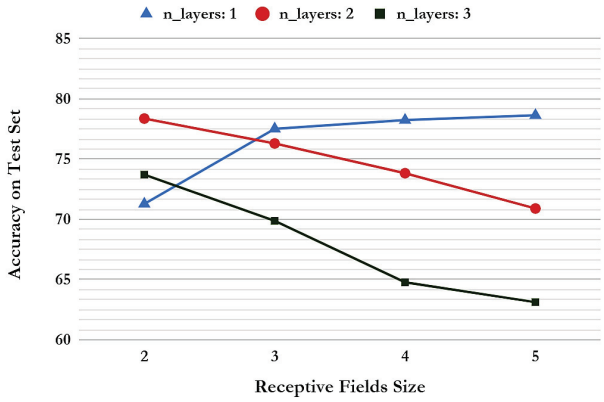

Figure 2: The performance of syntactic model across different receptive field sizes and number of layers(n_layers)

LSTM for POS encoding Figure 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 (Neal et al. 2017).

As shown in Table 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

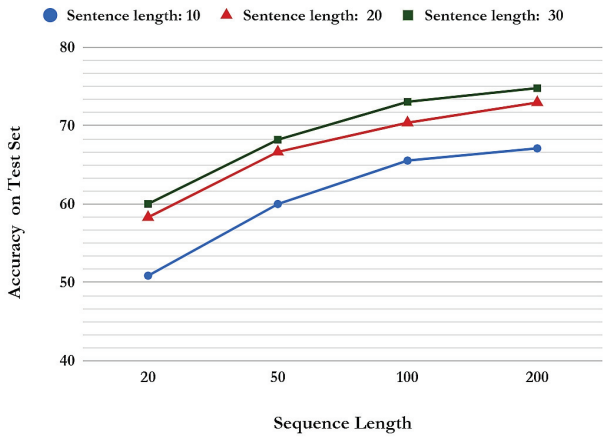

Figure 3: The performance of syntactic model across different sentence lengths and sequence lengths

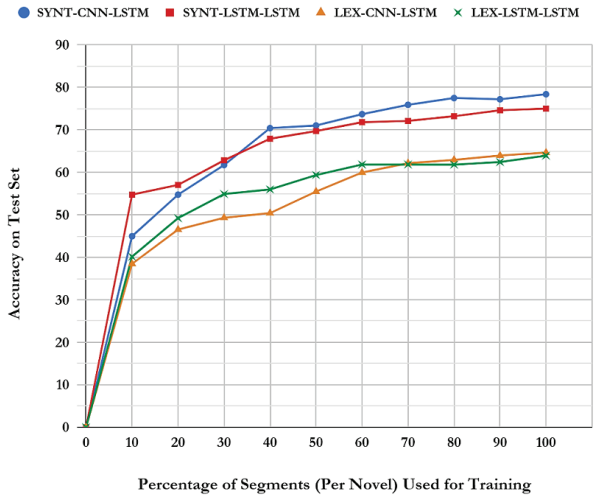

Figure 4: The performance of syntactic models when trained on different number of segments per novel

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.

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 classified independently and afterwards the label of each document is calculated as the majority voting of its constituent segments. Table 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 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 time 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 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 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.

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-

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)

Table 2: The performance results of models on PAN 2012 dataset for authorship attribution task.

LSTM models. This property, in particular, makes them a preferred potential model when investigating authorial writing style in a dataset of short documents.

Conclusion and Future Work

In this paper, we introduced 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 document for style-based text classification. 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 which this property makes them a preferable model when investigating authorial writing style in a dataset of short documents.

Acknowledgments

This work was funded by Crystal Photonics Inc (CPI) under Grant Number 1063271.

References

Argamon-Engelson, S.; Koppel, M.; and Avneri, G. 1998. Style-based text categorization: What newspaper am i reading. In *Proc. of the AAAI Workshop on Text Categorization*, 1–4.

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

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

Daelemans, W. 2013. Explanation in computational stylometry. In *International Conference on Intelligent Text Processing and Computational Linguistics*, 451–462. Springer.

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

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

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

Krause, M. 2014. 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*, 201–202. ACM.

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

Mosteller, F., and Wallace, D. 1964. Inference and disputed authorship: The federalist.

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

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

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

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

Shrestha, P.; Sierra, S.; Gonzalez, F.; Montes, M.; Rosso, P.; and Solorio, T. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, 669–674.

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

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