Contribution of Improved Character Embedding and Latent Posting Styles to Authorship Attribution of Short Texts

Wenjing HUANG † and Mizuho IWAIHARA ‡

Graduate School of Information, Production and Systems, Waseda University

2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka, 808-0135 Japan

E-mail: † huangwj_wendy@akane.waseda.jp, ‡ iwaihara@waseda.jp

Abstract Text contents generated by social networking platforms tend to be short. The problem of authorship attribution on short texts is to determine the author of a given collection of short posts, which is more challenging than that on long texts. Considering the textual characteristics of sparsity and using informal terms, we propose a method of learning text representations using a mixture of words and character n-grams, as input to the architecture of several neural networks. We also focus on the implicit characteristics of posts and incorporate them into the models. Our experimental evaluations on tweets show a significant improvement over baselines.

Keyword authorship attribution, short texts, social network platforms, character n-grams, CNN, LSTM, latent posting styles

1. Introduction

As online social activities become active, massive short texts are generated over social networking platforms. Classification on short text is a research hotspot following classification on long texts, but the former is more challenging than the latter. Author-labeled text classification, also known as Authorship Attribution (AA), is a fundamental branch in text classification. The task of AA is intended to identify the authors of given texts. The problem of AA on short texts has stimulated growing interest along with the explosion of social network traffic [12]. The AA system can be incorporated into application scenarios of detecting multiple IDs of a unique user, filtering spams [8] and avoiding identity frauds [12].

The core of solving the AA problem is to capture writing styles of target authors, which is relatively easy to achieve for long texts but restricted by sparse features in short texts. Several classification models implemented on AA, such as SVM [12], CNN [13] and RNN [2], have achieved certain success and demonstrated outstanding performance of word n-grams and character n-grams in discriminating the writing styles of authors. Schwartz et al. [12] feed word representations with several character n-grams and word n-grams into a SVM classifier. Shrestha et al. [13] employ a sequence of character n-grams as input to the CNN model. Inspired by the theory of word and character n-grams, we propose a method of applying improved character embedding on neural networks, that is, to embed a sequence of character n-grams mixed with special words into typical neural networks, such as CNN, LSTM, etc.

Twitter, as one of the most popular social networking platforms, provides a vast space for users to share individual ideas. Twitter stipulated that the length of tweets was no more than 140 characters by 2017, allowing users to express the core content in a very short space. We expect to capture users' latent posting styles from the short posts according to the underlying characteristics exhibited by the posts. Although the restricted length of the tweets is short enough, the actual length of the tweets varies from author to author. Social networking platforms, including Twitter, provide special functions that users can mention others (e.g., @Jack) and join topics (e.g., #Titanic). Emoticons are also popular among Twitter users. Additionally, certain users tend to use URLs, numbers, time and dates more often than others.

Apart from the writing characteristics of the posts, the sentiment tendency expressed by authors in the posts is a concern [8]. Most previous work indicates that users are used to expressing positive, neutral or negative opinions [6]. For example, "*How charming Jack is*", "*Jack gave the chance of survival to Rose*" and "*I cry for this sad story*" respectively voice the above three categories of sentiment. Another focus of opinion mining is subjective and objective expressions in posts. For example, "*Jack hits my heart*" and "*Jack is dead but love will go on*" respectively represents these two expressions.

In this paper, we introduce an additional feature set with 10 elements: text length, number of @<username>, number of #<topic>, number of emoticons, number of

URLs, number of numeric expressions, number of time expressions, number of date expressions, polarity-level and subjectivity-level. We also employ neural network models with tuned hyperparameters on the feature set to capture their feature expressions. Therefore, one neural network accepts text sequences as input for capturing textual features, and another neural network accepts posting-style sequences for capturing combinations of latent features. Then we concatenate the two generated vector representations and feed the combination to the softmax layer for author identification.

Text classification approaches based on traditional machine learning are usually calculating TF-IDF scores and training classification models. So we adopt a method of calculating TF-IDF scores and training a logistic regression classifier as one of our baselines in experiments. In addition, we refer to the main experimental results in [13] as other baselines. Then we evaluate the effectiveness of our proposed neural network models which incorporate improved character embedding and additional latent posting styles. Moreover, we discuss the performance of our models in more difficult scenarios, that is, when the number of authors or the number of posts per author varies.

The remainder of this paper is organized as follows. The related work is introduced in Section 2. Section 3 describes our two proposed methods on AA problem, including improved character embedding and latent posting styles. Section 4 presents our experimental details and evaluation results. Concluding remarks are discussed in Section 5.

2. Related Work

The large coverage of social networks has led to increasing research efforts on content generated by social media. AA researches have been gradually extended on web data such as emails [1], forums [14], and blogs [4]. [3] provides an in-depth analysis of author attribution in social media. When it comes to shorter texts, the existing methods for AA are difficult to achieve similar performance compared to long texts [10]. Word n-grams and character n-grams are widely used in existing AA methods [5][8][12][13][15], since they can capture syntactical features of the texts. For low-dimensional vector representation of posts, most of previous work is based on word embedding over certain special word n-grams and character n-grams [8][12]. Also, [13] is based on character embedding with character n-grams. There is no precedent work that is based on character embedding with mixed words and character n-grams. With the development of deep learning methods, both CNNs [9][13][11] and RNNs [2] have been applied in AA problem, showing outstanding performance. Especially, the effect of character n-grams applied on the CNN model is remarkable. LSTM is a variant of RNN, which can deal with the shortcomings of RNN in processing long sequences. LSTM has been successfully applied in text classification [16][17]. The method of applying character n-grams on LSTMs also performs competitively [13].

Features hidden in posts can also be utilized for AA. Post authors' sentiment orientations are one of the important latent characteristics [8]. Other features, such as text length, number of user mentions, number of topic mentions, and number of URLs, also help to characterize authors' writing styles [8].

In this paper, we propose approaches of applying improved character embedding to neural network models, and introducing latent posting styles as extended features to the architecture.

3. Methodology

In this section, we describe our proposed CNN and LSTM-based models for AA utilizing improved character embedding and latent posting styles.

3.1 Improved Character Embedding Method

Our proposed model is inspired by N-gram CNN [13] that combines character n-grams and CNN. We propose a method of applying improved character embeddings to neural networks such as CNN and LSTM.

Character n-grams. The character n-gram method has a remarkable performance in previous work on AA of short texts [13]. It has been observed that social networking platforms often emerge with informal terms. The character n-gram method can tolerate misspellings and informal usages of punctuation [15]. For example, the character bigrams of "nooooooo" are represented as "no" and "oo", which restore the form of the term "no". Let us consider another example of emoticons composed of punctuations. The character bigrams of emoticons ":-)" and ":-(" are respectively represented as ":-", "-)" and ":-", "-(", although the two emoticons have the same component ":-", the different components "-)" and "-(" hide the key sentiments of the emoticons. The character n-gram method can also extend the original short word-level sentence into a longer character-level sentence, which improves the sparseness of short text to a certain extent.

Improved Character Embedding. We observe that users frequently use mentions @<username> and hashtags #<topic> on social networking platforms such as Twitter. Schwartz et al. [12] replace all the forms of mentions @<username> with the same tag, ignoring the information of the user groups followed by the authors. However, our method retains the characteristics of user reference information, since we believe the same users mentioned frequently in posts will help identify authorship. Similarly, topic references are useful features. In our method, we keep all forms @<username> and #<topic> from being split by the character n-grams method. Therefore, we obtain sequences of mixed words and character n-grams. Table 1 shows examples of the mixed words and character bigrams. First, texts are transformed into lowercase. Considering that values of URLs, numbers, time, dates are sparsely occurring in posts, we replace these values with the tags "U," "N," "T," "D," respectively.

Sentences	Mixture of Words and Character Bigrams
@rose you jump, i jump! #titanic	**@rose** yo ou u_ _j ju um mp p, ,_ _i i_ _j ju um mp p! **#titanic**
how to lose weight? U #health	ho ow w_ _t to o_ _l lo os se e_ _w we ei ig gh ht t? ?_ _U **#health**
report at T: temperature: N, daily rain: N #weather	re ep po or rt t_ _a at t_ _T T: :_ _t te em mp pe er ra at tu ur re e: :_ _N N, ,_ _d da ai il ly y_ _r ra ai in n: :_ _N **#weather**

Table 1: Examples of mixed words and character bigrams. We replace spaces in the sentences with "_".

Then we use Word2Vec's Skip-Gram model with window size 5 to pre-train 300-dimension word vectors on the training set which includes mixtures of words and character n-grams. The Skip-Gram model works better than the CBOW model in predicting words from experience. In the character embedding module, we use pre-trained word vectors to represent the mixed sequences of words and character n-grams. The dimension of the embedding matrix is set to 140 on Twitter datasets and sequences with a length shorter than 140 are padded.

Neural Network Models. We apply our improved character embedding method on typical neural networks, namely CNN and LSTM. Our proposed architecture receives a mixed sequence of words and character n-grams as input. Then we use neural network models to automatically extract textual features of the sequence and obtain a compact feature vector representation. Finally, we apply a fully connected module with softmax function to process the representation for author classification.

Figure 1 presents the adoption of the CNN model into this architecture. In the convolutional layer, we use three types of filters with different size w and n filters for each type. Then the convolution results representing text features are upstreamed to a pooling layer with a max-pooling function to extract the most important features. Finally, the representation from concatenated pooling outputs is passed to the fully connected layer.

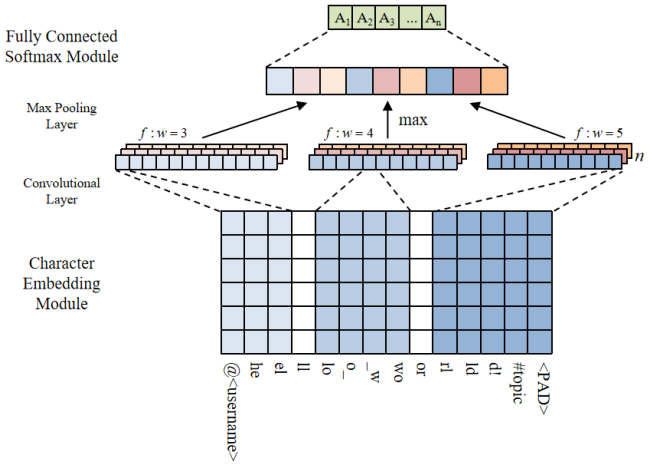

Figure 1: CNN model with improved character embedding. Mixed words and character n-grams are embedded to convolutional and max pooling layers, and the final representation is passed to a fully connected module with softmax function for classification.

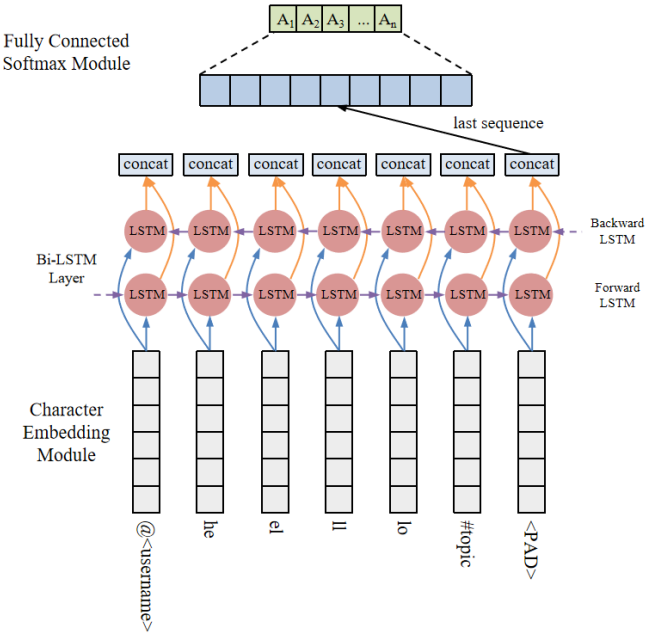

Figure 2: LSTM model with improved character embedding. Mixed words and character n-grams are embedded to Bi-LSTM layer, and the final output of the last time step is passed to a fully connected module with softmax function for classification.

The situation where a LSTM model replaces the CNN

module is presented in Figure 2. We adopt a two-layer bi-directional LSTM (Bi-LSTM) model to obtain the feature representation of an input sequence. Then we take the output of the last time step as the input to the fully connected module.

3.2 Latent Posting Styles

Most previous work focuses on textual features in AA tasks while very few explore latent features observed in posts. Authors' sentiment orientation and other posting expressions can help identify authors' writing styles, especially useful for AA of short texts [8]. We divide the post length into 5 levels (L1, L2, L3, L4, L5) ranging from 0 to 140. For the characteristics of using @<username>, #<topic>, URLs, numbers, time, dates and emoticons, we count their frequencies. Each post carries its author's sentiment, which may be positive/neutral/negative, and objective/subjective. We use polarity and subjectivity scores generated by TextBlob [7] for these abstract sentiments. Polarity scores vary from -1.0 to 1.0, where 1.0 is positive. Subjective score describes the degree of subjectivity of a post, which varies from 0 to 1.0. To incorporate sentiment characteristics into posting style vectors, we assign discrete levels P1, P2, P3, P4, P5 to polarity scores, and similarly assign discrete levels S1, S2, S3, S4, S5 to subjectivity scores. The tag representations for ten latent posting characteristics are shown in Table 2.

Features	Encoding	Tags
length	level	L1-L5
@<username>	count	'M'+count
#<topic>	count	'H'+count
URLs	count	'U'+count
numbers	count	'N'+count
time	count	'T'+count
dates	count	'D'+count
emoticons	count	'E'+count
polarity	level	P1-P5
subjectivity	level	S1-S5

Table 2: Tag representations for latent posting features.

All the latent features of posts are extracted to form a dataset with sequences of feature tags. Then we train a CNN or LSTM model with appropriate hyperparameters, using posting-style vectors pre-trained by Skip-Gram in the word embedding layer, to generate vector representations of posting styles. Finally, we concatenate these tag representations with the text representations obtained from the neural network models, as input to the fully connected softmax module. The overall system is depicted in Figure 3. It is worth noting that the neural network model used to capture the tag features is the same as that used to capture text representations. In our experiments, we compare combinations of (text, CNN) ⊕ (feature tags, CNN) and (text, LSTM) ⊕ (feature tags, LSTM), where each bracket represents (input sequence, neural network model).

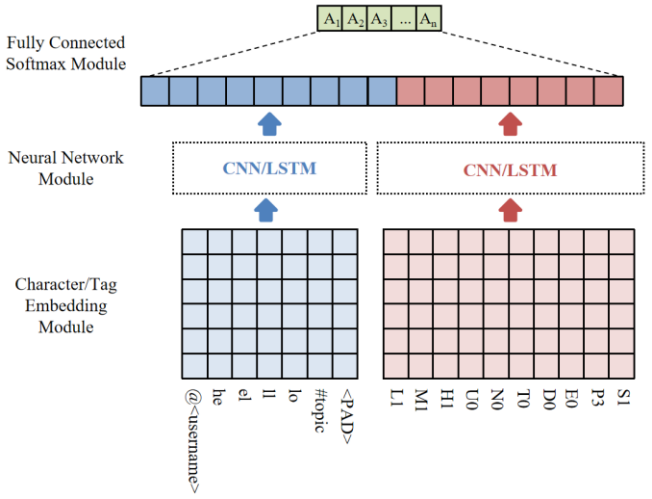

Figure 3: Proposed model with latent posting styles. Tag sequences are embedded by Skip-Gram into tag representations, which are passed to the CNN/LSTM model, and its output is concatenated with the output text representation from the left part, as input to the softmax layer.

4 Experiments

In order to verify the effectiveness of our methods, we utilize the Twitter dataset from Schwartz et al. [12], which contains groups of up to 9000 Twitter users with up to 1000 posts for each user, and approximately 9 million posts in total. We also adopt their experimental configurations. We employ 10-fold cross validation on all experiments.

Pre-Processing. Non-English tweets, tweets with less than three words and retweets have been already removed from the dataset. Considering sparsity, we replace URLs, numbers, dates and time with tags 'U', 'N', 'D' and 'T' respectively. Since @ and # may express different meanings in tweets, we distinguish mentions @<username> from occurrences of '@' in email addresses, emoticons ':@' and @ meant as 'at'. We also distinguish hashtags in the forms #<topic> from others.

Baselines. We construct a logistic regression classifier over TF-IDF scores of words as a baseline. We also refer to the experimental results of [13], which applies word-level or character-level word embeddings on CNN and LSTM models, for comparisons.

Our Models. We train CNN and LSTM models over word

vectors of mixed words and character n-grams (n = 1, 2, 3) which are pre-trained by Skip-Gram. We further incorporate embeddings of latent posting styles, and evaluate their effectiveness over the above models.

All the methods and descriptions used for our experimental evaluations are listed in Table 3.

Methods			Descriptions
Baselines		TF-IDF+LR	Traditional machine learning-based, calculate TF-IDF scores for words, then train a logistic regression classifier.
		CNN-W	Train a CNN model over word embeddings. [13]
		CNN-1	Train a CNN model over embeddings of character n-grams (n = 1, 2). [13]
		CNN-2	
		LSTM-2	[13] evaluates an LSTM trained on bigrams. LSTM has been successfully applied in text classification [16].
Our Models	CNN-based models	CNN-WC1	Word vectors of mixed words and character n-grams (n = 1, 2, 3) pre-trained by Skip-Gram are supplied to the character embedding layer of the CNN model.
		CNN-WC2	
		CNN-WC3	
		CNN-WC1+LPS	Combinations of latent posting styles (LPS) with CNN-WC1, CNN-WC2, and CNN-WC3.
		CNN-WC2+LPS	
		CNN-WC3+LPS	
	LSTM-based models	LSTM-WC1	Word vectors of mixed words and character n-grams (n = 1, 2, 3) pre-trained by Skip-Gram are supplied to the character embedding layer of LSTM.
		LSTM-WC2	
		LSTM-WC3	
		LSTM-WC1+LPS	Combinations of latent posting styles (LPS) with LSTM-WC1, LSTM-WC2, and LSTM-WC3.
		LSTM-WC2+LPS	
		LSTM-WC3+LPS	

Table 3: Overall methods and their descriptions. WC: mixed words and character n-grams. LPS: latent posting styles.

4.1 Experimental Details

In the stage of model construction, we experimentally set the best combination of hyperparameters for the CNN and LSTM models, by implementing the two models on a small set for hyperparameter tuning. The details of the hyperparameter combinations for the CNN and LSTM models are shown in Table 4.

Layer	# of Layers	Hyperparameters	
Embedding	1	length	140
		dimension	300
CNN	3	filter_sizes	[3, 4, 5]
		num_filters	[128, 128, 128]
		pooling	max
LSTM	2	architecture	bi-directional
		hidden_dim	128
Fully Connected	1	# of units	Depends on the number of authors

Table 4: Hyperparameter details of CNN and LSTM models.

In extended experiments, we integrate latent posting styles trained by CNN or LSTM model with original neural network architecture. The details of hyperparameter settings for training latent posting styles are shown in Table 5.

Layer	# of Layers	Hyperparameters	
Embedding	1	length	10
		dimension	100
CNN	2	filter_sizes	[2, 3]
		num_filters	[64, 64]
		pooling	max
LSTM	2	architecture	bi-directional
		hidden_dim	64
Fully Connected	1	# of units	Depends on the number of authors

Table 5: Hyperparameter details of training latent posting styles.

In addition, we add a dropout layer with keep_prob of 0.5 to the CNN and LSTM models to prevent the models from overfitting. We set the batch_size of 64 to process the data in batches for speeding up the model training process. We set a learning rate of 1e-3 and a learning rate decay of 0.9 to help the model converge while training. Besides, we introduce gradient clipping with the threshold set to 6.0 to solve the problem of gradient explosion. We limit the training epoch to 100 and screen out the best models with the minimum validation error.

4.2 Basic Results

First, we randomly select 10 groups of datasets containing 50 users and their 1000 tweets each. We evaluate the average accuracy of the models on the 10 groups of datasets with cross validation on training sets. The experimental results are shown in Table 6.

Methods			Accuracy
Baselines		TF-IDF+LR	0.674
		CNN-W	0.548
		CNN-1	0.757
		CNN-2	0.761
		LSTM-2	0.645
Our Models	CNN-based models	CNN-WC1	0.815
		CNN-WC2	0.828
		CNN-WC3	0.798
		CNN-WC1+LPS	0.824
		CNN-WC2+LPS	**0.836**
		CNN-WC3+LPS	0.806
	LSTM-based models	LSTM-WC1	0.717
		LSTM-WC2	0.739
		LSTM-WC3	0.701
		LSTM-WC1+LPS	0.744
		LSTM-WC2+LPS	0.762
		LSTM-WC3+LPS	0.725

Table 6: Accuracy for 50 authors with 1000 tweets each.

From the results of the baselines, we can observe that although the traditional machine learning method

(TF-IDF+LR) can achieve a good accuracy of 0.674, the deep learning methods have greater potential for improvement. For the deep learning models, the CNN models far surpass the LSTM models in performance with the same character n-grams embeddings. On the other hand, the CNN models with character n-grams embeddings far outperform those with word embeddings. In the baseline system, CNN-2 achieves the best accuracy of 0.761, which is the state-of-the-art result on this dataset.

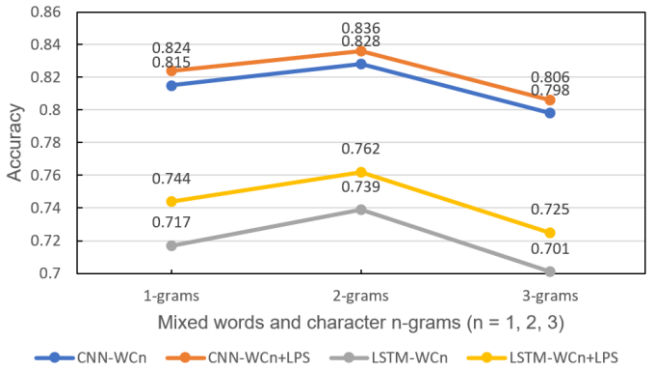

Figure 4: Accuracy comparison among our models. WCn: mixed words and character n-grams (n = 1, 2, 3). LPS: latent posting styles.

Our proposed models have two core improvements: improved character embedding and latent posting styles, which contribute to significant improvements over the baselines. CNN-WC2+LPS model shows the best performance with an accuracy of 0.836, exceeding CNN-2 by 7.5%. Figure 4 illustrates the performance of the CNN and LSTM models using mixed words and character n-grams (WCn, n = 1, 2, 3) and latent posting styles (LPS), with n-grams as the abscissa and accuracy as the ordinate. Our method of applying embeddings of mixed words and character n-grams on neural networks achieves the maximum performance when n = 2. In addition, the CNN-based models are far superior to the LSTM-based models. On the other hand, the method of introducing latent posting styles can effectively improve the models and the effect is more significant on LSTM-based models, which has about 2.5% improvement, while about 0.8% improvement on CNNs. Figure 5 shows the superiorities of our models over the baselines (CNN-1, CNN-2 and LSTM-2). Obviously, our improved character n-grams embedding method outperforms the existing character n-grams embedding method. It also shows that adding additional latent posting styles helps model optimization.

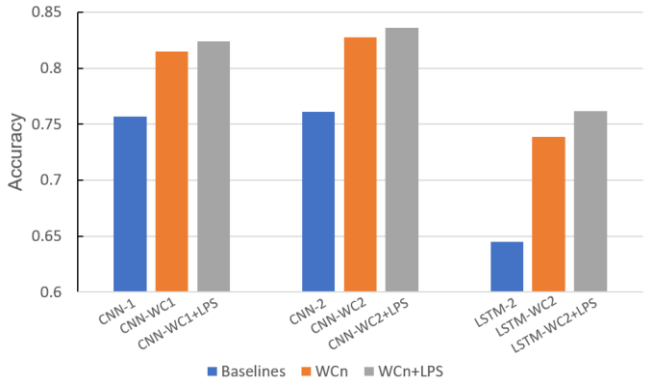

Figure 5: Accuracy comparison of our models with the baselines (CNN-1, CNN-2 and LSTM-2). WCn: mixed words and character n-grams (n = 1, 2, 3). LPS: latent posting styles.

4.3 Varying Numbers of Authors

To verify the effectiveness of our methods, we further explore the performance of our models in more difficult scenarios, as Schwartz et al. did [12]. One of the scenarios is when the number of authors changes with the same number of tweets per author. We conduct several series of experiments using the same selected groups of 100, 200, 500, 1000 authors and 200 tweets each. Considering that the method of mixed words and character n-grams performs obviously better when n = 1, 2 than when n = 3, we omit experiments when n = 3. The experimental results are presented in Table 7. The increase in the number of authors makes the task of authorship attribution more difficult. Figure 6 also illustrates this situation. Nevertheless, Our CNN-based models (see yellow marks in Figure 6) still have clear advantages over all baselines, and our LSTM-based models (see green marks in Figure 6) are still better than LSTM-2 and CNN-W. When the number of authors reaches 1000, CNN-WC2+LPS model even obtains an accuracy of 0.510, which is a 12.6% improvement over the best baseline.

Methods			Number of Authors			
			100	200	500	1000
Baselines		TF-IDF+LR	0.454	0.453	0.411	0.384
		CNN-W	0.241	0.208	0.161	0.127
		CNN-1	0.508	0.473	0.417	0.359
		CNN-2	0.506	0.481	0.422	0.365
		LSTM-2	0.338	0.335	0.298	0.248
Our Models	CNNs	CNN-WC1	0.580	0.556	0.509	0.453
		CNN-WC2	0.598	0.590	0.555	0.489
		CNN-WC1+LPS	0.609	0.590	0.551	0.508
		CNN-WC2+LPS	**0.616**	**0.599**	**0.564**	**0.510**
	LSTMs	LSTM-WC1	0.414	0.338	0.308	0.253
		LSTM-WC2	0.428	0.357	0.324	0.259
		LSTM-WC1+LPS	0.435	0.358	0.321	0.265
		LSTM-WC2+LPS	0.458	0.386	0.339	0.288

Table 7: Accuracy for varying numbers of authors with 200 tweets each.

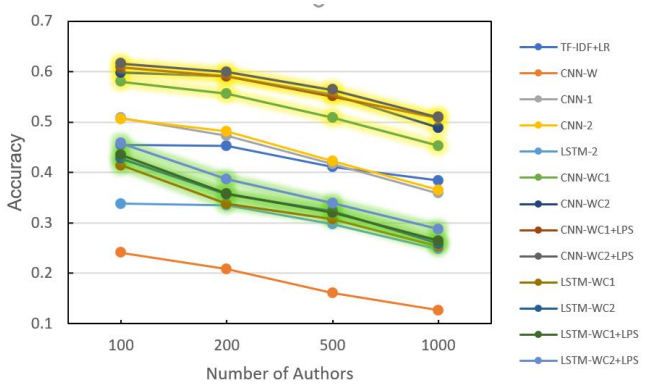

Figure 6: Accuracy comparison with the number of authors increasing.

Methods			Number of Tweets			
			500	200	100	50
Baselines		TF-IDF+LR	0.614	0.551	0.486	0.372
		CNN-W	0.509	0.460	0.417	0.366
		CNN-1	0.717	0.665	0.617	0.562
		CNN-2	0.724	0.665	0.613	0.542
		LSTM-2	0.597	0.528	0.438	0.364
Our Models	CNNs	CNN-WC1	0.772	0.690	0.610	0.572
		CNN-WC2	0.770	0.677	0.550	0.460
		CNN-WC1+LPS	0.783	0.696	**0.617**	**0.581**
		CNN-WC2+LPS	**0.792**	**0.701**	0.554	0.471
	LSTMs	LSTM-WC1	0.667	0.588	0.492	0.387
		LSTM-WC2	0.686	0.609	0.513	0.401
		LSTM-WC1+LPS	0.674	0.598	0.506	0.396
		LSTM-WC2+LPS	0.690	0.617	0.523	0.409

Table 8: Accuracy for varying numbers of tweets under 50 authors.

4.4 Varying Numbers of Tweets

Another scenario is when the number of tweets changes under the same number of authors. We evaluate our models on groups of the dataset with 50 authors and their 50, 100, 200, 500 tweets each. The results when varying numbers of tweets are shown in Table 8. When the number of tweets per author decreases, the number of training samples for each author decreases, making the classification task more difficult. The results in Figure 7 follows this trend. Considering that all the datasets are too small, we refer to the method of Shrestha et al. [13], which takes the average accuracy of the experiments on 10 disjoint datasets as the results. From the evaluation results, we can conclude that our CNN-based models perform more stable when n = 1 (see red marks in Figure 7) than when n = 2 (see blue marks in Figure 7), which is consistent with one of the findings in [13]. We have noticed that when the number of tweets per author is no more than 100, CNN-WC2 and CNN-WC2+LPS models perform worse than CNN-2. This is because our methods are more dependent on information, so they cannot show obvious advantages on very small datasets. Besides, our LSTM-based models (see green marks in Figure 7) perform better than baselines except CNN-1 and CNN-2.

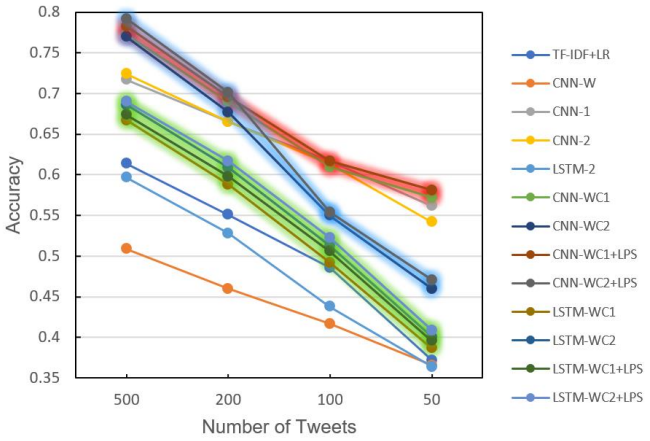

Figure 7: Accuracy comparison with the number of tweets per author decreasing.

5 Conclusions

This paper discussed new approaches for authorship attribution on short texts. The superior performance of the convolutional neural network with character n-grams embeddings has inspired us to propose new improved methods, using mixed words and character n-grams, instead of just character n-grams. We set up two sets of comparative experiments to test our ideas on CNNs and LSTMs. Rigorous experiments prove that our methods show clear advantages for solving AA problems on short texts. In addition, we capture ten latent posting styles for each tweet and use the corresponding neural network to train posting-style vectors, which are then integrated with the network architecture. The introduction of latent posting styles shows different performance improvements in the CNN and LSTM models, which is about 0.8% improvement in the former and about 2.5% improvement in the latter. Our best method achieves an accuracy of 83.6%, which is 7.5% improvement over the state-of-the-art result. Furthermore, as the number of authors increases or the number of samples per author decreases, the AA tasks become more difficult. Nevertheless, our models have clear advantages.

References

[1] A. Abbasi and H. Chen, “Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace,” ACM Transactions on Information Systems (TOIS), 26(2):7, pp. 17-29, 2008.

[2] D. Bagnall, “Author identification using

multi-headed recurrent neural network," In Working Notes Papers of the CLEF 2015 Evaluation Labs, Vol. 1391.

[3] M. Koppel and Y. Winter, "Determining if two documents are written by the same author," Journal of the Association for Information Science and Technology, 65(1), pp. 178–187, 2014.

[4] M. Koppel, J. Schler, and S. Argamon, "Authorship attribution in the wild," Language Resources and Evaluation, 45(1), pp. 83–94, 2011.

[5] R. Layton, P. Watters, R. Dazeley, "Authorship Attribution for Twitter in 140 characters or Less", IEEE Proc. 2nd Cybercrime and Trustworthy Computing Workshop, CTC' 10, Washington D.C., A, pp. 1-8, 2010.

[6] Y. Lin, X. Wang, and A. Zhou, "Opinion Analysis for Online Reviews," World Scientific, Vol. 4, 2016.

[7] U. Malik, "Python for NLP: Introduction to the TextBlob Library," Stack Abuse, April 15, 2019. [Online]. Available: https://stackabuse.com/python-for-nlp-introduction-to-the-textblob-library/. [Accessed April 15, 2019].

[8] L. Patamawadee and M. Iwaihara, "Utilizing Latent Posting Style for Authorship Attribution on Short Texts," IEEE CBDCom 2019, Fukuoka, pp.1015-1022, Aug. 2019.

[9] D. Rhodes, "Author attribution with cnns," 2015. [online]. Avaiable: https://www. semanticscholar. org/paper/Author-Attribution-with-Cnn-s-Rhodes/0a904f9d6b47dfc574f681f4d3b41bd840871b6f/pdf. [Accessed on Aug. 22, 2016].

[10] A. Rocha, et al. "Authorship Attribution for Social Media Forensics," IEEE Trans. Info. Forensics and Security, Vol.12, No. 1, Jan. 2017.

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

[12] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel, "Authorship Attribution of Micro-Messages," Proc. 2013 Conf. Empirical Methods in Natural Language Processing, Seattle, pp. 1880-1891, Oct. 2013.

[13] P. Shrestha, S. Sierra, F. A. Gonzalez, P. Posso, M. Montes-y-Gomex, and T. Solorio, "Convolutional Neural Networks for Authorship Attribution of Short Texts," Proc. 15th Conf. European Chapter of the Assoc. Computational Linguistics, Vol. 2, Valencia, pp. 669-674, Apr. 2017.

[14] T. Solorio, S. Pillay, S. Raghavan, and M. Montes-Gomez, "Modality specific meta features for authorship attribution in web forum posts," Proc. 5th Conf. International Joint on Natural Language Processing, pp. 156-164, Nov. 2011.

[15] E. Stamatatos, "A survey of modern authorship attribution methods," Journal of the American Society for Information Science and Technology, 60(3), pp. 538–556, 2009.

[16] K.S. Tai, R. Socher, and C.D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," arXiv preprint arXiv: 1503.00075, 2015.

[17] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," Proc. 2015 Conf. Empirical Methods in Natural Language Processing, pp. 1422-1432, 2015.