Convolutional Neural Networks for Authorship Attribution of Short Texts

Prasha Shrestha
Dept. of Computer Science
University of Houston
Houston, TX, 77004
pshrestha3@uh.edu

Sebastian Sierra and **Fabio A. González**
Computing Systems and
Industrial Engineering Dept.
Universidad Nacional de Colombia
Bogotá, Colombia
{ssierral, fagonzalezo}@unal.edu.co

Paolo Rosso
Universitat Politècnica
de València
Valencia, Spain
prosso@dsic.upv.es

Manuel Montes-y-Gómez
Instituto Nacional de
Astrofisica, Optica y Electronica
Puebla, Mexico
mmontesg@ccc.inoep.mx

Thamar Solorio
Dept. of Computer Science
University of Houston
Houston, TX, 77004
solorio@cs.uh.edu

Abstract

We present a model to perform authorship attribution of tweets using Convolutional Neural Networks (CNNs) over character n-grams. We also present a strategy that improves model interpretability by estimating the importance of input text fragments in the predicted classification. The experimental evaluation shows that text CNNs perform competitively and are able to outperform previous methods.

1 Introduction

The problem of authorship attribution (AA) has always been harder for short texts compared to long texts. Previous work has shown that it is difficult for any AA system to maintain the same performance with shorter texts (Koppel and Winter, 2014). However in today's world where most human interaction is online and short, AA of short texts has become ever more relevant, especially in areas like phishing emails, spam, and crowd sourced collaborative projects like Wikipedia. With the advent of social media, one can even argue that building systems that work with short texts equally, if not more important than long texts like books. This need is also reflected in the increasing interest in AA of small texts such as tweets and reviews in AA research community (Qian et al., 2015; Schwartz et al., 2013; Layton et al., 2010).

At the time of this writing, we could neither find any prior work that successfully applied character n-grams with CNNs, nor any CNN methods that dealt with AA of short text. However, we were able to find research in AA using traditional as well as related approaches. Character and word n-grams have been used as the core of many authorship attribution systems (Stamatatos, 2009; Schwartz et al., 2013; Layton et al., 2010). Character and word n-grams help determine the author of a document by capturing the syntax and style of an author. Considering deep learning approaches, we found one other work that uses CNNs for authorship attribution (Rhodes, 2015). However, they use word representations for larger texts rather than character representation for short texts. Additionally, work by Bagnall (2015) uses a multi-headed Recurrent Neural Network (RNN) character language model that gives a set of next character probabilities for each author at every step of the model. This was the best-performing system for the PAN 2015 author identification task with a macro-averaged area under the curve (AUC) of 0.628 (Stamatatos et al., 2015). Despite the promising results that CNNs and RNNs show, the results are not interpretable and few of these works attempt to analyze what the networks are actually learning. We try to get an insight into our model by using the saliency analysis by Li et al. (2016). We have also devised our own method of finding out the input n-grams that are overall most important to the model.

As a solution to the problem of AA of short texts, we propose a neural network architecture that is able to learn the representation of the text starting from the character sequence. Our architecture is a CNN that uses a sequence of character n-grams as input. This contrasts with the tradi-

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 669–674,
Valencia, Spain, April 3-7, 2017.

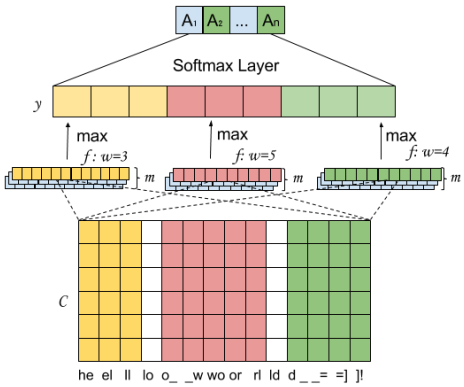

Figure 1: **N-gram CNN**. N-gram embeddings are fed to convolutional and max pooling layers, and the final classification is done via a softmax layer applied to the final text representation (_: whitespace in the input).

tional approach to CNN that uses either a sequence of words or a sequence of characters (Kalchbrenner et al., 2014; Kim, 2014; Collobert et al., 2011; Zhang et al., 2015). This CNN captures local interactions at the character level, which are then aggregated to learn high-level patterns for modeling the style of an author. The main contributions of this paper are:

- We are the first to present a CNN model based on character n-grams for AA of short texts. We also show a comparison with traditional machine learning approaches.
- We validate the robustness of our model against traditional AA architectures by evaluating it in different settings.
- We propose a new method to improve interpretability of our CNN model.

2 N-gram Convolutional Neural Networks

Our proposed architecture receives a sequence of character n-grams as input. These n-grams are then processed by three modules: a character embedding module, a convolutional module, and a fully connected softmax module, as illustrated in Figure 1. Our character embedding module is motivated by the success of other distributed vector representations like word embeddings (Mikolov et al., 2013) This module learns a continuous, non-sparse d-dimensional vector representation of the character n-grams. The maximum length l of the training sequences determines the size of the input and input shorter than l are padded. This module yields a matrix $C \in \mathbb{R}^{d \times l}$, where the columns are the embedding of the n-gram c_j of position j.

The next component is a convolutional module. First a convolution filter, $H \in \mathbb{R}^{d \times w}$, is applied to a portion of C, where w is the width of the filter. The resulting matrix, O, is used as input to a sigmoid function g, along with a bias term b to produce feature representations f for the text.

$$O = H \cdot C[i : i + w - 1]$$

$$f = g(H \cdot C[i : i + w - 1] + b), f \in \mathbb{R}^{l-w+1}$$

As can be seen from Figure 1, we use a convolutional layer with different widths w, allowing us to capture patterns that involve everything from morphemes to words. We then pool the resulting feature maps f by max-over-time pooling (Collobert et al., 2011), to obtain y_k, the maximum value of each feature map f_k:

$$y_k = \max_i f_k[i], k = 1 \dots m$$

where m is the number of feature maps. This allows us to represent the text by its most important features, independent of their position. After pooling and concatenating the feature representations y_k, we obtain a compact representation of the text.

Finally, this representation is passed through a fully connected module containing a softmax layer. Representation learning models based on neural networks attempt to find features that are useful to solve a learning problem automatically. In the case of AA, stylistic features may be found at morphological, lexical and syntactic levels. We hypothesize that our model is able to automatically capture patterns at all these levels by starting at short sequences of characters and then using convolution to generate representations for longer sequences.

2.1 Implementation details

Layer	# of layers	Hyperparameters	
Embedding	1	l	140
		d	300
Convolutional	3	m	[500, 500, 500]
		w	[3, 4, 5]
		Pooling	max
Fully connected	1	# of units	Depends on the # of authors

Table 1: Neural network architecture hyperparameters

Table 1 contains the combination of hyperparameters for the three modules that generate the best validation score. Additionally, we have added

a dropout layer with 25% dropout after the first embedding layer for regularization. We then shuffle and group the samples into mini-batches of size 32 for faster training. We employ Adaptive Moment Estimation (Kingma and Ba, 2015) with a learning rate of $1e-4$ to train our network. We train for a maximum of 100 epochs and choose the model with the lowest validation error.

3 Experimental Evaluation

CNN-2	CNN-1	SCH	CHAR	LSTM-2	CNN-W
0.761	0.757	0.712	0.703	0.645	0.548

Table 2: Accuracy for 50 authors with 1000 tweets each.

We evaluated our approach on the dataset from Schwartz et al. (2013) containing ∼9,000 Twitter users with up to 1,000 tweets each, using the same train/test splits, and normalized URLs, usernames, and numbers. We trained separate CNN models with character n-grams ($n = 1, 2, 3$) on a small validation set. Here we evaluate our two best-performing models, one on unigrams (**CNN-1**) and another on bigrams (**CNN-2**), against three other systems described below:

SCH: The Schwartz et al. (2013) work uses character 4-grams and word 2-5 grams. They also introduced k-signatures and flexible patterns to represent the unique signature of an author. Their best system uses a combination of all these features.

LSTM-2: Long Short Term Memory networks (LSTM) have been successfully used for text classification (Tai et al., 2015; Tang et al., 2015). We evaluate an LSTM trained on bigrams, since the LSTM produced better results on a small validation set.

CHAR: Character and word n-grams have been the core of many AA systems (Stamatatos, 2009; Schwartz et al., 2013; Layton et al., 2010). We tested various n-gram combinations on the small validation set and our final system uses character 2,3,4-grams with logistic regression.

CNN-W: Many works on CNN use word sequences as input (Kalchbrenner et al., 2014; Rhodes, 2015). We also trained a CNN model with Google Word embeddings (Mikolov et al., 2013) fed to a static embedding layer.

All systems use cross-validation over the training set for hyperparameter tuning. We first experimented with a relatively small set of 50 authors and their 1000 tweets each. The results are in Table 2. The results show that our CNN bigram model (CNN-2) performs very well on this dataset and outperforms the SCH system by nearly 5%. CNN-1 also exceeds the SCH method but is marginally worse than CNN-2, showing that there is merit in exploring the training of a CNN model on n-grams rather than only on single characters.

3.1 Varying number of authors and tweets

# of authors	CNN-2	CNN-1	SCH	CHAR	LSTM-2	CNN-W
100	0.506	**0.508**	0.425	0.412	0.338	0.241
200	**0.481**	0.473	0.411	0.409	0.335	0.208
500	**0.422**	0.417	0.355	0.342	0.298	0.161
1000	**0.365**	0.359	0.303	0.291	0.248	0.127

Table 3: Accuracy comparison for increasing # of authors with 200 tweets per author.

We also wanted to explore how our method fares against the other methods when the problem becomes more difficult, i.e. when the number of authors increases or when the number of tweets per author decreases, as done in Schwartz et al. (2013). The results for increasing number of authors are shown in Table 3. Both our CNN models perform fairly well above the other methods for all our experiments. Although the accuracy decreases with the increasing number of authors, even with 1000 authors our model obtains an accuracy well above 36%, and there is a 6% improvement over the state-of-the-art (SCH).

# of tweets	CNN-2	CNN-1	SCH	CHAR	LSTM-2	CNN-W
500	**0.724**	0.717	0.672	0.655	0.597	0.509
200	**0.665**	**0.665**	0.614	0.585	0.528	0.460
100	0.613	**0.617**	0.565	0.517	0.438	0.417
50	0.542	**0.562**	0.507	0.466	0.364	0.366

Table 4: Accuracy comparison for decreasing # of tweets per author for 50 authors.

We can draw similar conclusions from the results where we decrease the number of tweets per author as shown in Table 4. Following the work in SCH, these results are an average of the accuracy values obtained from 10 disjoint datasets. The performance of our system is fairly stable even when the number of tweets per author is low. The improvement margin actually increases slightly as we move towards a lower number of tweets.

A statistical t-test on the results over the 10 disjoint datasets shows that the difference between CNN-2 and CHAR, LSTM-2, and CNN-W are statistically significant at $p < 0.001$. We could not perform a test with SCH results as the individual disjoint dataset results are not reported. In both these tables, we can see that the CNN-2

CNN-2	CNN-1	CHAR	LSTM-2	CNN-W
0.683	0.678	0.609	0.525	0.420

Table 5: Accuracy values for 35 authors with 1000 tweets each after bot-like authors removal (15 authors were bots).

model outperforms the CNN-1 model for experiments with more data points (higher no. of authors and/or tweets), which can be attributed to CNN-2 having a higher number of parameters to train. CNN-W performs worse than the other systems. Char-based inputs specialize on stylistic patterns whereas word-based ones focus on content-related patterns, which are less important for AA. This finding is consistent with previous research in AA (Stamatatos, 2009; Koppel and Winter, 2014; Koppel and Schler, 2004).

3.2 Bot-like Authors

During analysis, we noticed that nearly 30% of authors behave like automated bots. Their tweets show repeated patterns, e.g., a title of some news/advertisements with a URL at the end. Since our goal is to perform AA on humans, we removed these authors manually to create a refined dataset. There are no comparable experiments in (Schwartz et al., 2013), thus we compare only against CHAR, LSTM-2, and CNN-W as shown in Table 5. The accuracies for all of the methods decrease on this dataset as the bot-like authors are easy to identify. The CNN methods still outperform other methods. Since SCH's performance was similar to CHAR on the whole dataset and CNN-2 exceeds CHAR by a larger margin in this dataset, we can estimate that here too, CNN-2 is likely to outperform SCH.

4 What does the CNN capture?

Despite the competitive performance of neural representation techniques in several NLP tasks, there is a lack of understanding about exactly what these models are learning, or how the parameters relate to the input data. Few empirical studies have attempted to understand the role of RNN components (Jozefowicz et al., 2015; Greff et al., 2016). In order to analyze what makes neural representation learning suitable for AA, we look at the most salient sections of a single input tweet. We also perform an analysis of what types of character n-grams are more important to the model overall.

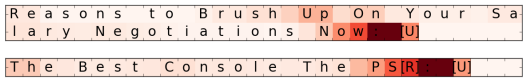

Figure 2: Salient sections of a bot-like author's tweets ([U]:URL, [N]:username, [R]:number).

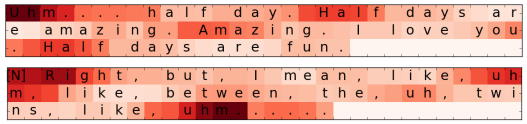

Figure 3: Salient sections of a human author's tweets ([U]:URL, [N]:username, [R]:number).

Salient sections of a tweet

Li et al. (2016) define a saliency score $S(e)$ as:

$$w(e) = \frac{\partial(S_c)}{\partial(e)} \qquad S(e) = |w(e)|$$

where the embedding e represents the input and the class score S_c represents the output of our CNN model. The score indicates how sensitive a model is to the changes in the embedding input, i.e. in our case, how much a specific n-gram in the text input contributes to the final decision. In order to visualize saliency per character, we adapted this method by taking the maximum saliency value per character.

We selected two authors, one bot-like and one human, to analyze what kind of patterns are learned for specific authors. Figure 2 presents two tweets from a bot author. The darker the shade is, the more salient that section of the tweet is in the attribution decision. This automated bot seems to follow the pattern *Title: URL* and sure enough, it is detected by the CNN-2 model as indicated by dark shading towards the end of both tweets. Similarly, Figure 3 shows two tweets from a human author. We can notice right away that this author has the tendency to use *uhm* and we can see this section highlighted in the figure. The author also tends to use consecutive dots, this too is highlighted, albeit a little less than *uhm*. Figure 4 shows the saliency values for a tweet from the CNN-2 (top) and the CNN-1 (mid) models.

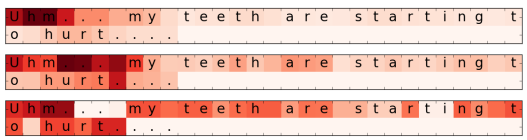

Figure 4: Salient sections comparison of CNN-2 (top) and CNN-1 (mid). The bottom figure is shaded using the feature weights from logistic regression for CHAR.

Dataset	Highest activations overall	Top activations per filter	CHAR top features
Bot & non-bot	_[U], Di, :l, n‘, (:, Xn, KM, o), :_, =h, _[R], :-, qh, wu, !,	bi, ul, al, ug, me, in, mp, AN, um, an, en, "w, sa, e_	:_[U], :, _–, _u, _r, .._-, _X, XD, _XD, li, go,_., _#
Non-bot only	_[U], qh, KM, Di, (:, Uh, ;D, :p, _[N], __, :l, !_, =D, :_	_t, _m, er, ou, e_, in, ed, co, _a, is, nd, _r, ve, te, st	...,;-), lol, :d, maoo, &&, :)), :-(, :-p, loll, ????, ^_^

Table 6: Input char bigrams with highest CNN activations ([U]:URL, [N]:username, [R]:number, _:whitespace).

For the CNN-1 model, although *uhm* and ... are highlighted, the saliency values are more distributed throughout the tweet, highlighting even *are* and *hurt*. While we can see that the CNN-2 model puts its focus exactly on the *uhm*, which is a very distinctive style of this author. Figure 4 also has a similar figure for the CHAR model at the bottom, which we created by using the feature weights from the logistic regression classifier. Although there is more focus on the *uhm* part, again, the distribution is more spread out for this model as well, compared to the CNN-2 model.

N-grams with highest contributions Some n-grams activate several filters, but generate low activation values, meanwhile, other n-grams generate higher activation values but only for a few filters. Both types hold important clues in understanding our model. We use the intermediate representation of the CNN filters, consisting of a matrix $O \in \mathbb{R}^{n \times m}$ where n is the number of n-grams and m is the number of filters. We first determine the n-grams that generate the highest activation values aggregated over all filters. The second column in Table 6 shows the top 15 bigrams from this analysis for CNN-2 models trained on the whole dataset and on the refined dataset. The third column presents the top positive weighted features from the CHAR model. We can observe that many of the highest bigrams are uncommon versions of emoticons, such as *(:*, *:p* and *;D* that are likely correlated with specific authors. For the bot authors, *[U]* has the highest activation since most automated tweets have URLs at the end as their characteristic.

We then also collect the n-grams that have the highest number of filters where their activation is in the top 3. The third column in Table 6 shows the top bigrams from this analysis. Here we mostly see bigrams that are affixes. We can attribute this fact to the importance of morphological features for characterizing human tweets.

5 Conclusions and Future work

We presented a strategy for using CNNs with character n-grams for AA of short texts, and provided a comprehensive comparison against standard approaches. We found that CNNs give better performance for AA of tweets, and using character n-grams instead of just character sequences can also improve performance. We were also able to gain some insights on what our architecture is actually learning. We could see that the network is focusing more on some sections of the text. This creates a premise for applying attention models and we are currently working in this direction.

Acknowledgements

This research is partially supported by NSF award number 1462141, Colciencias Research Grant FP44842-576-2014, CONACYT-Mexico (247870), and SomEMBED MINECO TIN2015-71147-C2-1-P. We want to thank the anonymous reviewers and Simon Tice for their invaluable suggestions.

References

Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*, volume 1391.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, PP(99):1–11.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the*

2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Moshe Koppel and Jonathan Schler. 2004. Authorship verification as a one-class classification problem. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 62–, New York, NY, USA. ACM.

Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65(1):178–187.

Robert Layton, Paul Watters, and Richard Dazeley. 2010. Authorship attribution for twitter in 140 characters or less. In *Proceedings of the 2010 Second Cybercrime and Trustworthy Computing Workshop*, CTC '10, pages 1–8, Washington, DC, USA. IEEE Computer Society.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, June. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR), Workshop*.

Tie-Yun Qian, Bing Liu, Qing Li, and Jianfeng Si. 2015. Review authorship attribution in a similarity space. *Journal of Computer Science and Technology*, 30(1):200–213.

Dylan Rhodes. 2015. Author Attribution with CNN's. `https://pdfs.semanticscholar.org/0a90/4f9d6b47dfc574f681f4d3b41bd840871b6f.pdf`.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891, Seattle, Washington, USA, October. Association for Computational Linguistics.

Efstathios Stamatatos, Martin Potthast, Francisco Rangel, Paolo Rosso, and Benno Stein. 2015. Overview of the pan/clef 2015 evaluation lab. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 518–538. Springer.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal, September. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.