Classification of Micro-Texts Using Sub-Word Embeddings

Mihir Joshi
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
mihir@dal.ca

Nur Zincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
zincir@cs.dal.ca

Abstract

Extracting features and writing styles from short text messages is always a challenge. Short messages, like tweets, do not have enough data to perform statistical authorship attribution. Besides, the vocabulary used in these texts is sometimes improvised or misspelled. Therefore, in this paper, we propose combining four feature extraction techniques namely character n-grams, word n-grams, Flexible Patterns and a new sub-word embedding using the skip-gram model. Our system uses a Multi-Layer Perceptron to utilize these features from tweets to analyze short text messages. This proposed system achieves 85% accuracy, which is a considerable improvement over previous systems.

1 Introduction

One of the most challenging problems in text analysis is identifying the author of a micro-text, i.e. short text messages. Most of the data created on social media applications whether a Tweet, Facebook comment or text on a messaging application is a micro text. A micro or short text message could be a tweet or a comment which is around 140 characters or less. In general, text analysis and natural language processing approaches employ statistical feature extraction techniques such as term frequency, inverse document frequency and a bag of words. Due to the feature extraction process being statistical in nature, all these techniques require a certain amount of data to use them effectively for determining patterns or perform authorship attribution. Thus, having short text messages such as tweets which is around 140 characters or less makes it difficult to identify the patterns on a given text and make predictions about the author.

Shrestha et al. (2017) used a convolutional neural network (CNN) architecture using character embeddings instead of word embeddings for short texts. With this approach they showed less than five percent improvement on previous researches (Qian et al., 2015; Schwartz et al., 2013) in this area. In this paper, we start by implementing a simpler approach; combining the two consecutive records from the same author to train our model and then apply the feature known as Flexible patterns (Schwartz et al., 2013) after modifying the existing method and finally introducing our new feature based on the word vector approach (Bojanowski et al., 2017).

The paper is organized in the following sections. Section 2 represents related works. In section 3 we define our model architecture followed by the dataset and the feature extraction techniques. Section 4 shows the results of our approach compared with previous works. Lastly, in section 5, we discuss conclusion and future work.

2 Related Works

In short-text analysis, one of the earlier works by Layton et al. (2010) aims to identify the author based on the data collected from micro-blogging websites like Twitter. The authors create author profiles using character level n-grams. They find the frequency of the most common n-grams in an author profile and assume that text from the same author would have a similar pattern. This approach is further extended by Schwartz et al. (2013); they use two more features namely word n-grams and a Hyponyms acquisition technique (Hearst, 1992) called Flexible patterns along with character n-grams. They then input a combination of these features into a linear SVM and ten-fold cross validation is applied to evaluate the model.

Proceedings of Recent Advances in Natural Language Processing, pages 526–533,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_062

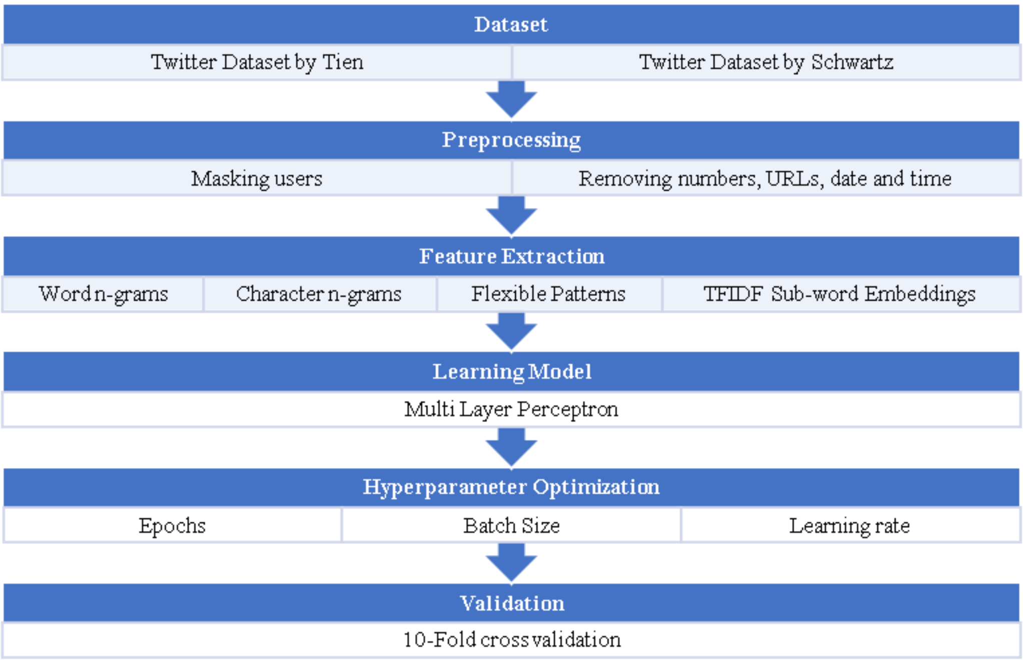

Figure 1: Overview of the proposed system

One of the first deep learning approaches by Rhodes (2015) used word embeddings and a convolutional neural network. Skip-gram method with negative sampling is used for word vectorization from the Google news dataset while using random initialization for unseen words. Furthermore, the convolutional neural network was based on a similar model (Collobert et al., 2011) for language processing, where the activation function is changed to rectified linear units and introducing dropouts. Finally, it demonstrated that the system performs well on longer text sequences.

Shrestha et al. (2017) makes use of a similar architecture; a convolutional neural network (CNN) using character embeddings instead of word embeddings for short texts. The CNN model takes character unigram or bigram as an input that passes through the character embedding layer before feeding it to the convolutional layer. Similar datasets and evaluation criteria as Schwartz et al. (2013) are used to determine the performance of the system. This approach increases the overall accuracy to approximately 76%. Even though the results are better than all previous researches there is a scope for further improvement.

A more recent study on this field (Phan and Zincir-Heywood, 2018) used word embeddings and the neural network to identify the authors of short text messages. They used the three different datasets namely Reuters Corpora (RCVI) (Lewis et al., 2004), Enron dataset (Klimt and Yang, 2004), and Twitter dataset (Yilu et al., 2016). The RCVI is used to train the embeddings which in turn generated the high-level features from the other two corpora by concatenating the vector of means and standard deviations. They then used the feed forward neural network to identify the authors.Therefor, in this work, we also used the same five authors (*Ashley_Nunn75, bradshaw1984, shawnevans81 , terrymarvin63, and WhieRose65*) from the twitter dataset that they used to test our proposed system.

3 Methodology

In this section, we describe our proposed approach and the features we used to train our system. Figure 1 shows an overview of our system in which we worked on two datasets: one by Schwartz et al. (2013) and the other one by Yilu et al. (2016). Section 3.1 explains more about the data.

We carried our pre-processing on the data before preforming feature extraction. There are a total of four features that are obtained from the text namely word n-grams, character n-grams, flexible patterns and word embeddings. An n-gram (section 3.2) is a sequence of n words where n is a positive integer. For example, in the phrase - *"This is a sentence"* -, a word unigram would be - *"This", "is", "a"* - and a word bigram would be *"this is", "is a", "a sentence"* - and so on. On the other hand, character n-grams are similar to word n-grams except, they are a sequence of characters. For instance, if we use the previous example, ”This is a sentence”, a character unigram would be - *”T”, ”h”, ”i”, ”s”* - and a character bigram would be - *”th”, ”hi”, ”is”*.

In section 3.3 we define Flexible patterns, explain the existing method and our new approach to create them. The idea behind flexible patterns is that some users tend to use the same sequence of words in their writing style and only change a few keywords called content words (CW). For example, the flexible pattern of the following phrases; *”I read the paper today”* and *”I drove the car yesterday”* is *”I CW the CW CW”*. The words *read, paper, today, drove, car* and *yesterday* are replaced by the word *CW* based on the pre-defined condition. Therefore, masking some words, would create a pattern and separate the texts of some users from other users. We modified the existing approach (Schwartz et al., 2013) to make it suitable for smaller datasets and also to make it easier to implement.

Next, we talked about the word embeddings (Mikolov et al., 2013) feature set in section 3.4. We used the skip-gram technique to create the 300-dimensional vector representation of our data and used that to create the embeddings by combining them using the weights obtained by using TF-IDF.

Later we analyzed (section 3.5) all the models we used and the reason for choosing a Multi-Layer Perceptron (MLP) architecture (Gillian, 2014). We also explained the architecture of our MLP model and the parameters we used to further optimize. Lastly, in section 4 we will compare the results of our approach to both datasets.

3.1 Datasets

To compare our results with the previously mentioned approaches, we used the same dataset that Schwartz et al. (2013) used. The dataset contains a total of 7000 authors, out of which we selected 50 authors at random, where each author has 1000 tweets. We masked the username (@user), numbers, links, date and time from the texts to minimize the bias and noise in the data. We also changed all of the letter characters to lowercase and employ word stemming to reduce the size of the vocabulary used.

Before extracting features and feeding data into the MLP, we concatenated sets of two adjacent records. For example, the first text is combined with the second, the third with the fourth and so forth. Though the initial number of records remains the same, combining the original texts enables us to have a larger sequence, which achieves better accuracy even with the earlier approaches of feature extraction (Schwartz et al., 2013; Shrestha et al., 2017). The larger sequence also helps us to achieve more meaningful patterns which is not otherwise possible.

We also applied the above approach on a different dataset (Yilu et al., 2016) used in Phan and Zincir-Heywood (2018). We used the same 5 authors (users) as they did, with 2000 tweets each. In that paper (Phan and Zincir-Heywood, 2018), the names of the tagged users in a tweet were not masked. However, we masked them, i.e. whenever we encounter a tagged user - *@user* - we change it to a specific word preventing our system from overfitting.

3.2 Word and Character N-Grams

Both word and character n-grams were extracted from the datasets used. For the value of *n* in n-grams and the minimum occurrence of each n-gram, we used the same parameters as used by Schwartz et al. (2013) for comparison purposes. We also took into consideration the maximum occurrence of the n-grams. For example, a pattern including prepositions (as shown in 1 below) or a masked username (as shown in 2 below) are very common in tweets.

1. *for a*
2. *<User> I*

This is even more common in character n-grams. For this reason, we kept the upper limit of the n-grams to 0.9, which means we do not consider any word or character n-grams that appear in more than 90 percent of the documents. This further helps us identify the unique writing style of an author.

We restricted each feature to a maximum of 50,000 in our experiments where each author had 1000 tweets. To assign weights to the n-grams, we used the term frequency-inverse document frequency (TF-IDF) weighted scheme.

TF-IDF (Manning et al., 2008) calculates the importance of the word based on how frequently it appeared in a text, which was then balanced by the frequency of the word in the entire corpus.

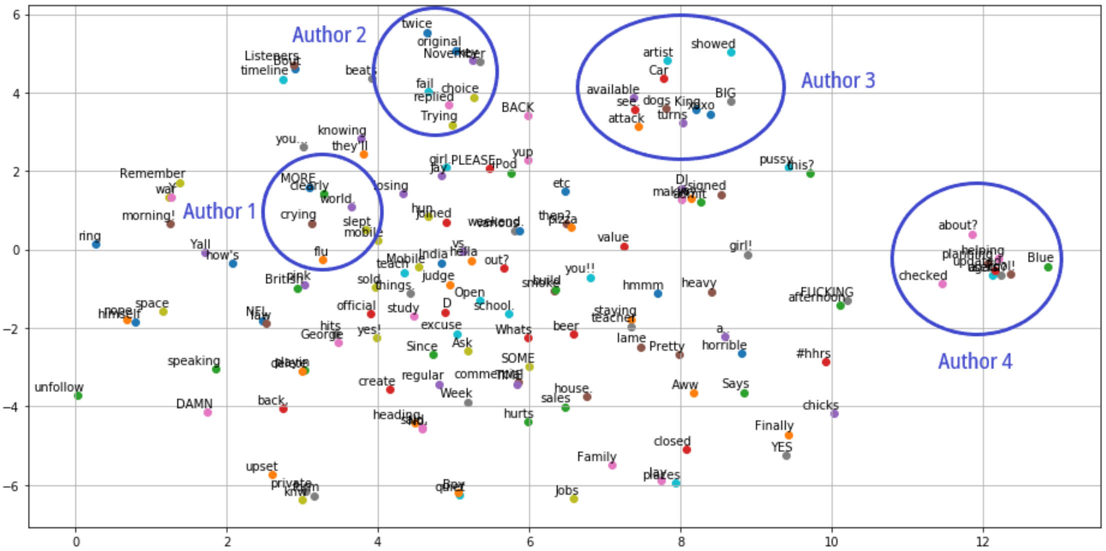

Figure 2: Skip-Gram word embeddings

In a text document d, the frequency of term t is: $tf_{t,d} = f_{t,d}$. This shows the total number of times (frequency count) the term occurs in the document, whereas the inverse document frequency is: $idf(_{t,d}) = log\frac{N}{t \in D}$, where $\widehat{\text{N}}$ is the total number of documents D in the corpus. Moreover, we employed sub-linear scaling to the TF-IDF by taking the log of the term frequency, Eq.1.

$$wf_{t,d} = \begin{cases} 1 + logtf_{td}, & \text{if } tf_{td} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

3.3 Flexible Patterns

Flexible patterns are a branch of word n-grams, where each word is either a high-frequency word or a content word and some words can be both (Schwartz et al., 2013). For a corpus size s if a word appears more than $10^{-4} \times s$ times it is a High-Frequency Word (HFW) and if it appears less than $10^{-3} \times s$ times it is a Common Word (CW). Also, the previous method takes into consideration that flexible patterns start/end with an HFW and there can be no consecutive HFWs.

The problem with this approach is that it does not work with a smaller corpus. For a corpus where the value of s is 1000, no word is a CW. Thus, to overcome this limitation, Eq.2 is used to calculate the CW for a bigger corpus. Therefore in a corpus with a vocabulary size n, the CW is calculated as the common log of the threshold for CW is selected as twice of log of s which are total number of words in the training dataset vocabulary. This method is based on the various experiments we carried out to choose the optimum number.

$$CW \leqslant 2 \times log_{10}n \tag{2}$$

After replacing all the CWs in the corpus, we then used the same approach as we did for word n-grams. Then we applied the TF-IDF weighted scheme to those flexible n-grams before inputting them into the model.

3.4 TF-IDF Weighted Word Embeddings

Word embedding (Mikolov et al., 2013) is a semantic parsing technique used to create the vector representation of a text in a smaller dimensional space compared to the classic Bag of words approach (Zhang et al., 2010). The idea behind word embedding is that semantically similar words should be close to each other. That is, in an n-dimensional space, the angle between the similar words should be close to zero. There are two types of methods to achieve this; namely, Continuous Bag of Words (CBOW) and Skip-Gram methods (Mikolov et al., 2013). While CBOW predicts the target words by taking the context word as input, Skip-gram predicts the probability of the context words using the target words.

We used the word embedding approach from Bojanowski et al. (2017), which is based on the assumption that tweets contain several words, including, but not limited to, hashtags that are rare and sparsely occur in a corpus. To address this sparsity issue, each word is represented by the sum of the vector representations of its n-grams. Having a dictionary as size D and a word w belongs to this dictionary having $D_w \subset \{1, ..., D\}$, the set of n-grams appearing in w. Associating a vector representation of z_d to each n-gram d, the scoring function for w can be represented by Eq.3:

$$s(w, c) = \sum_{d \in D_w} z_d^T v_c \tag{3}$$

For the value of n, we chose $2 \leqslant n \leqslant 6$. Empirically, instead of using pre-trained embedding, we trained it on our corpus with a 300-dimension vector space. We constructed the embedding using the Skip Gram approach which works better for a smaller amount of data and is preferable when there are a greater number of rare words in the corpus (Schwartz et al., 2013). Figure 2 shows the created vector representation from the dataset.

Using t-distributed Stochastic Neighbor Embedding technique (Maaten and Hinton, 2008), we visualized the 300-dimensional data in a two-dimensional space (Figure 2). This shows that words which are used in the same context are grouped closer to each other. The idea behind this approach is that a particular author would use the same combination of words in his/her tweets and therefore, they should be close to each other.

Before using this as an input feature for our model, we weighted the embedding of each word in a text with the IDF value of that word. The resulting dimension is the same as the dimension of each word, which in our case is 300. Ultimately, we calculated the mean of the words to keep the dimensionality of the entire text the same as our vector space. Having a text of size T and words w where $T_w \subset \{1, ..., T\}$, the TF-IDF weighted embedding feature $f(w)$ of a word w_T is given by using Eq.4:

$$f(w) = \frac{\sum_{w \in T_w} idf(w_T) \times emb(w_T)}{T} \tag{4}$$

4 Experiments and Results

As shown in Figure 1, four different machine learning algorithms are employed at the learning phase of the proposed system in this work. We implemented SVM, Naive Bayes, Random Forest (Decision Trees) and MLP classifiers using the Scikit-Learn Python machine learning library (Pedregosa et al., 2011). As discussed earlier, the goal is to classify micro-text. In doing so, we aim to choose the most suitable classifier using the extracted features proposed in the previous section. To this end, we employed the same small dataset used in Phan and Zincir-Heywood (2018). Table 1 shows the 10-fold cross validation accuracy of the four classifiers on that dataset for the 5 authors previously mentioned, where 2000 tweets are used for each author. In this case, our system (Table 1) achieves more than 99% 10-fold cross-validation accuracy which is 15% more than the best result reported in Phan and Zincir-Heywood (2018). Moreover, Figure 3 (confusion matrix) demonstrates that these results are not biased to any specific author in the dataset used. Additionally, Table 2 shows how the 10-fold cross-validation accuracy improved the proposed MLP classifier as the different feature extraction techniques are used, where the first three columns show the accuracy of the combination of extraction techniques and the last column is the best test results given in Phan and Zincir-Heywood (2018).

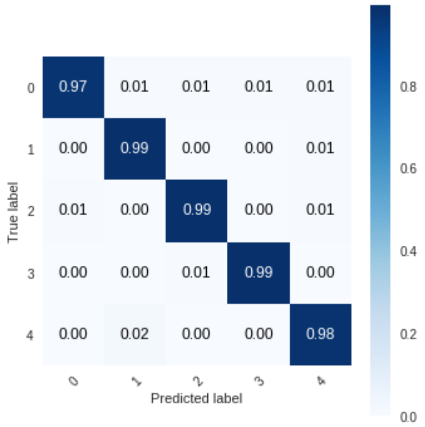

Figure 3: Confusion matrix for 5 authors

MLP	SVM	Naive Bayes	Random Forest
.99	.979	.96	.82

Table 1: Accuracy for 5 users with 2000 tweets using different classifiers having combination of same four features

Given the above observations, MLP was chosen as the most suitable classifier for our research purpose: classifying the tweets according to their

TFIDF Emb.	Mod. flex	Joining rec.	Phan2018
.99	.983	.972	.841

Table 2: Accuracy for 5 users with 2000 tweets each using proposed system with different feature sets

authors. Figure 4 presents the overview of the proposed MLP model. The number of nodes in the input layer depend on the size of the input feature. *None* represents that the dimension is variable, and in this case, it is *86121*. Then, there is a dense layer, which is a fully connected layer, where each input node is connected to each output node. In the proposed model, there are two Dense layers. One is a hidden layer of 1000 nodes and the other is the output layer of 50 nodes, which corresponds to 50 authors. This can be changed depending on the number of authors (output classes). Inbetween the two dense layers, there is a dropout layer for regularization.

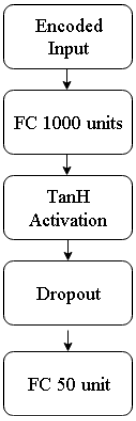

Figure 4: Overview of the proposed model using a Multi-Layer Perceptron as the classifier with one hidden layer and a dropout layer

The input layer of the MLP model depends on the number of features, which increases as the number of tweets increases to train the model. There is only one hidden layer with 1000 nodes. It should be noted, that increasing the hidden layers did not improve the validation accuracy in our experiments. The Tanh activation function (Weisstein, 2002) is used for the hidden layer. The last layer consists of the same number of nodes as the number of authors where the SoftMax activation function (Nwankpa et al., 2018) is used. Furthermore, we have a 30% dropout after the hidden layer to prevent overfitting. In the proposed system, ADAM (Kingma and Ba, 2014) is adapted as the optimization algorithm with a learning rate of 0.001. We divided the data into batches of 64 and trained our model for a maximum of 40 epochs. Figure 5 and Figure 6 show the number of epochs vs loss, and the number of epochs vs accuracy graphs for training and validation data, respectively. These results show that both the loss and accuracy are optimal at around 40 epochs.

In the following experiments, we incrementally

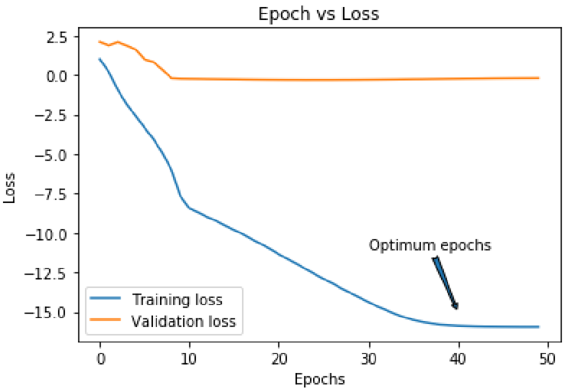

Figure 5: Epochs vs Loss for 50 epochs

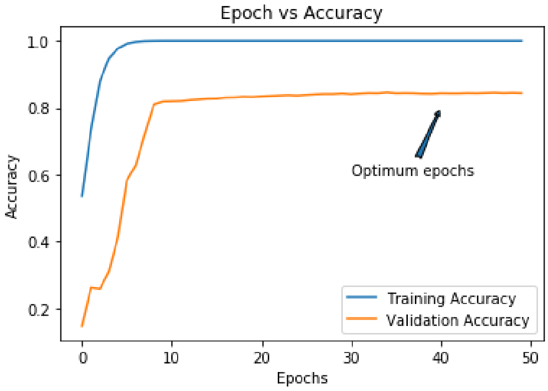

Figure 6: Epochs vs Accuracy for 50 epochs

TFIDF Emb.	Flex	Joining rec.	CNN-C	SCH	Char	LSTM-2
.852	.829	.81	.761	.712	.703	.645

Table 3: Accuracy for 50 users with 1000 tweets each

applied all three feature extraction techniques using the MLP classifier and observed the improvements compared to the previous research results (see Section 2). To this end, we first combined the subsequent tweets and applied the approach presented by Schwartz et al. (2013), then we improved that method to calculate flexible patterns and their effect on the accuracy of the system. Last but not the least, we built weighted TF-IDF embeddings and combined them with the improved flexible patterns to form the combined set of features as input into the proposed MLP model. Tenfold cross-validation approach was used to evaluate the performance of the proposed system. We evaluated the proposed system on the same dataset as used by Schwartz et al. (2013); Shrestha et al. (2017), where the dataset consisted of 50 authors, each having 1000 tweets.

Columns 1-3, in Table 3, shows the results of all three feature extraction techniques used with

#	TFIDF Emb.	Mod. flex	Joining rec.	CNN-C	SCH	Char	LSTM-2
500	**.791**	.76	.748	.724	.672	.655	.597
200	**.730**	.694	.679	.665	.614	.585	.528
100	**.648**	.619	.608	.617	.565	.517	.438
50	**.589**	.563	.53	.562	.505	.466	.364

Table 4: Accuracy for 50 users from 500 to 50 tweets for each

the MLP, and columns 4-7 represents the results from the previous works on the same dataset. Our results are mutually inclusive, and the results are built upon combining all the feature extraction techniques. For example, we used the weighted TF-IDF embeddings in combination with the flexible patterns.

CNN-C, shown in Table 3, represents the best result obtained by Shrestha et al. (2017) where a convolutional neural network architecture is proposed using character n-grams, specifically unigrams and bigrams as input. The convolutional model used was a three-level architecture with the input layer as the character embedding layer, a convolutional module, and finally, a dense layer with a Softmax activation function for classification. The unigram model performed well on the smaller dataset. Alternatively, the bigram model had better accuracy on the bigger dataset.

SCH, shown in Table 3, represents the best result obtained by Schwartz et al. (2013). They used a linear SVM for classification and their model was a combination of word and character n-grams along with a new feature set called flexible patterns (see section 3.3), which is modified and used in our system as well.

Char, shown in Table 3, represents one of the systems used by Shrestha et al. (2017) in which they compared the performance of their system based on the earlier character n-gram approaches (Layton et al., 2010; Schwartz et al., 2013) and proposed a logistic regression model that employed character n-grams of sizes two to four.

LSTM-2, shown in Table 3, represents the state-of-the-art LSTM model based on the success of previous implementations (Tai et al., 2015; Tang et al., 2015). This model was also used with bigrams as input to evaluate the performance of those systems, with respect to other models on the same dataset.

Introducing the concatenation of the consecutive records enables the accuracy of the proposed system to be improved by 5% compared to the previous approaches. Then applying the flexible patterns, further improved the accuracy by approximately 2%. Finally, implementing the weighted TF-IDF word embedding, and combining it with all the other features increases the accuracy of the proposed system to approximately 85%, Table 3.

In Table 4, we also compared the proposed system's accuracy to other approaches as we reduced the number of tweets from 500 to 50 for each author (50). After reducing the number of tweets, the proposed system still outperformed all the other previous approaches and performed well even when the dataset became as small as 50 tweets per author.

5 Conclusion and Future Work

In this paper, we explored a feature extraction technique that was based on a word embedding model weighted by TF-IDF. We also worked on modifying and improving the existing implementation of flexible patterns and proposed a neural network architecture that makes use of a combination of these features to perform authorship attribution. Our model outperformed all the existing systems based on similar testing criteria and using the same datasets. Since we trained our embeddings from scratch our system performs equally well, irrespective of the language.

With the success of word embeddings, we look forward to working upon new word embedding techniques such as Elmo (Peters et al., 2018) and Bert (Devlin et al., 2018), which are based on the context of a word in a corpus. Alongside that, we are also interested to improve our neural network architecture using transfer learning models such as ULMFit (Howard and Ruder, 2018).

6 Acknowledgments

This research is supported by the Natural Science and Engineering Research Council of Canada. This research is conducted as part of the Dalhousie NIMS Lab at: https://projects.cs.dal.ca/projectx/.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Nick Gillian. 2014. Mlp nickgillianwiki. http://www.nickgillian.com/wiki/pmwiki.php/GRT/MLP. (Accessed on 04/25/2019).

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 539–545.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*. Springer, pages 217–226.

Robert Layton, Paul Watters, and Richard Dazeley. 2010. Authorship attribution for twitter in 140 characters or less. In *2010 Second Cybercrime and Trustworthy Computing Workshop*. IEEE, pages 1–8.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5(Apr):361–397.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK. http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. 2018. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378* .

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research* 12:2825–2830.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Tien D Phan and Nur Zincir-Heywood. 2018. User identification via neural network based language models. *International Journal of Network Management* page https://doi.org/10.1002/nem.2049.

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 cnns cs224.

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.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. volume 2, pages 669–674.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

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.

Eric W Weisstein. 2002. Inverse hyperbolic tangent .

Zhou Yilu, Alsarkal Yaqoub, and Zhang Nan. 2016. Linking virtual and real-world identities twitter dataset.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1(1-4):43–52.