# Identifying Twitter Bots Using a Convolutional Neural Network

## Notebook for PAN at CLEF 2019

Michael Färber[1], Agon Qurdina[2], and Lule Ahmedi[2]

[1]Karlsruhe Institute of Technology (KIT), Germany
michael.faerber@kit.edu
[2]University of Prishtina, Kosovo
agon.qurdina@studentet.uni-pr.edu
lule.ahmedi@uni-pr.edu

**Abstract** In this paper, we present an approach for identifying Twitter bots based on their written tweets using a convolutional neural network. We experiment with various embedding methods (pretrained and trained on the training dataset) and convolutional neural network architectures and compare their performance. When evaluating our best performing approach on the actual test data set of the CLEF 2019 Bots Profiling Subtask (English language), we obtain an accuracy of 90.34%. We therefore see convolutional neural networks as a promising machine learning technique for Twitter bot detection.

## 1 Introduction

With over 1 billion active users, Twitter is among the most frequently used social media platforms. It has taken an important role as a medium disseminating information and opinions. However, due to its potential to influence the reader's opinion, bot nets are considered to be a threat to society and democracy. For instance, it has been reported that botnets have been identified in the context of public voting, such as the Brexit voting [9] and the federal elections of the United States in 2016 [6]. Botnets have also been used for making the ideologies of terrorism attractive [5]. It is estimated that about 15% of all active accounts on Twitter (i.e., 48 million) are bot accounts [14]. Developing approaches to detect Twitter bots automatically is therefore important.

In this paper, we consider the following task proposed as a shared task at CLEF 2019: "Given a Twitter feed, determine whether its author is a bot or a human." [4,13] In the past, several approaches to Twitter bot identification have been proposed (see Section 2). However, most of them do not use the content of the tweets, but instead the metadata of the tweets and their implications (such as the publication frequency over time). We present an approach based on convolutional neural networks where all tweets of a user are used as input.

This paper is structured as follows. In Section 2, we present related works on bot identification. In Section 3, we describe our approach for Twitter bot identification. In Section 4, we present the evaluation setup, the evaluation data set, and the evaluation results. We conclude in Section 5 with a summary and an outlook.

## 2 Related Work

Given that millions of user accounts on social media platforms are bots, classifying user accounts in social media has already been approached in various regards. In 2016, Adewole et al. [1] listed 65 articles about malicious account detection approaches for social networks, including social tagging and other social network variations. We differentiate between approaches that are non-content based and content-based.

*Non-content based approaches to bot identification.* Non-content-based approaches use patterns in the behaviour of the Twitter accounts. Chavosi et al. [3], for instance, perform correlation analysis, exploiting the fact that highly synchronous cross-user activities indicate whether an account is a bot or not. In [2], Beskow and Carley state that measured differences between bot and human conversation networks can be used to increase the accuracy in bot detection. Mazza et al. [12], one of the most recent works on bot detection, use patterns of the retweeting activity, with the specific goal of detecting malicious retweeting bots. Recently, Lundberg et al. [11] use only tweets' metadata in order to provide a language-independent approach to bot detection.

*Content-based approaches to bot identification.* Content-based approaches target the classification of Twitter accounts based on the account's tweets. In [10], the authors use content-based features and pattern contrast features for bot detection. They obtain classification results over 0.90 of AUC with this approach.

*Hybrid approaches to bot identification.* In their paper [8], Kudugunta and Ferrara propose a hybrid approach that exploits both tweet content and metadata (e.g., retweet and reply count, or number of hashtags) to detect whether a given tweet was posted by a human or a bot. The framework uses a deep neural network based on a contextual long short-term memory (LSTM) architecture and exhibits promising performance of over 0.96 of AUC to bot detection at the tweet-level.

Note that in this paper, we obtain an AUC score of 0.86 based solely on the content of the tweets. To the best of our knowledge, we are the first ones approaching such a task using a convolutional neural network model.

## 3 Approach

In this section, we present our approach for twitter bot identification. The source code of our implementation is available online at `https://github.com/agon-qurdina/author-profiling`.

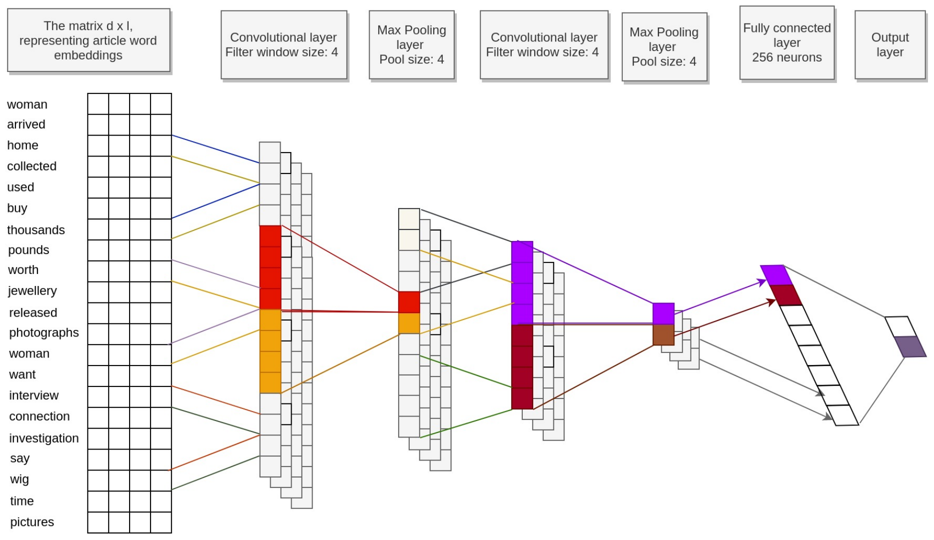
The matrix d x l, representing article word embeddings
Convolutional layer Filter window size: 4
Max Pooling layer Pool size: 4
Convolutional layer Filter window size: 4
Max Pooling layer Pool size: 4
Fully connected layer 256 neurons
Output layer
woman
arrived
home
collected
used
buy
thousands
pounds
worth
jewellery
released
photographs
woman
want
interview
connection
investigation
say
wig
time
pictures

**Figure 1.** Architecture of our system used for classifying sentences.

## 3.1 Preprocessing

Normally, the input for our model would be the set of tweets for a given user. But we decided to have the entire user's feed of tweets as a single input to the model. Thus, we merged all of the author's tweets together into a single block of text which we called "article." Next, given a set of "articles" as input, we preprocessed them along the following steps:

**Text Cleaning.** New lines were replaced by spaces. We also expanded contractions and removed stop words, HTML tags, and special characters from the articles' content.

**Texts to Sequences.** The articles' content was tokenized and a word dictionary (size: 155,227 unique words) was generated.

**Sequence Padding.** We applied a post-padding with a fixed sequence length $l$. Due to the way we concatenated the tweets to a single "article," we were left with a limited number of training samples. Thus, in order to keep as much information as possible, we set $l = 11000$. By choosing this high value for the sequence length, only a small percentage (284 from 2873) of the sequences were truncated.

## 3.2 Basic Model Architecture

Our basic architecture is shown in the Figure 1. We use a CNN architecture that is based on Kim et al.'s approach for sentence classification [7]. His proposed architecture has been widely applied for various tasks in the past. The CNN consists of two one-dimensional convolutional neural networks layers, followed by MaxPooling layers, with a dense neural networks layer processing the output of the second CNN layer. The model is completed by a final output layer that uses the sigmoid activation function to return a binary output (namely, the classification into *human* or *bot*).

In the following, we describe the architecture in more detail:

**Input layer.** Considering the article's words were embedded into $d$-dimensional vectors, the final matrix used as input to the model can be written as $I = l \times d$ where $l$ is the chosen sequence length (i.e., the length of the articles). Recall that $l = 11000$ in our setting.

**First convolutional layer.** The first transformation this embedded input goes through is a convolutional layer with $f$ 1-dimensional filters of length $k$. Thus, the layer weights can be considered a matrix of shape $Wc \in R^{f \times k}$.

We used a filter size $f$ of 64 and a filter length $k$ of 4. In our context, having 1-dimensional filters means that for each word in an article, its three adjacent words are considered as the context of the word. The output of the convolutional layer then is $C = conv(I, Wc)$ where $conv$ is the convolutional operation applied to input $I$ using the weights matrix $Wc$. This operation includes applying the *ReLu* activation function to complete weights calculations. Also, a dropout function is used to prevent overfitting. We used a dropout rate of $0.5$, which means $50\%$ of the weights (randomly chosen) during each training epoch are set to $0$.

**First max pooling layer.** The above output $C$ is then considered the input to a 1-dimensional Max Pooling layer. The purpose of the layer is to extract only the most important features of the convolution outputs. This is done by keeping only the max value from a pool size $p$. As we chose $p = 4$, the output of this layer can be written as $M = max\_pool(C, p)$. This operation reduces the number of weights by four times.

**Second convolutional layer and max pooling layer.** The output $M$ of the max pooling layer is the input of the second convolutional layer, and the whole process described above is applied to this input, to get a final output $M_2$.

**Fully connected layer.** Given the two-dimensional matrix of weights from the last step, the next layer in the network is a fully connected one with a size of $256$ hidden neurons. But in order for the convolutional output to serve as an input to this layer, its dimensionality needs to be reduced. This was done using the flatten method, which keeps all of the values but flatten them in a long vector. A *ReLU* activation function and a dropout layer with a rate of $0.5$ were used here.

**Output layer.** The last layer is a fully connected layer with one neuron. The sigmoid activation function is used to provide a binary output.

## 3.3 Architecture Variations

We developed and evaluated the following architecture variations:

1. The first variation uses 1-dimensional MaxPooling layers of size *2* after each of the Convolutional layers. That means the resulting number of features, after passing from the convolutional layer to the MaxPooling layer and then out of it, will be halved. Considering we start with a sequence length of 11,000, the fully-connected layer contains more than 11 million weights to train. We will call this variation Large Model in the remaining part of this paper.
2. The second variation uses 1-dimensional MaxPooling layers of size *4* after each of the Convolutional layers, which reduces the number of features generated by the Convolutional layers by a factor of 4. It does that by only keeping the max value

**Table 1.** Characteristics of the used evaluation data set.

| Dataset | Number of articles | In % |
|---|---|---|
| Train | 2873 | 69.85 |
| Validation | 1240 | 30.15 |

**Table 2.** Hyperparameters.

| Parameter | Value |
|---|---|
| Conv. filter size | 64 |
| Conv. kernel size | 4 |
| MaxPooling1D pool size | 4 |
| Dropout rate | 0.5 |
| Dense layer units | 256 |
| Layers Activation function | ReLu |
| Optimizer | Adam |
| Learning Rate | Adaptive (0.001$\rightarrow$0.00001) |
| Loss function | Binary crossentropy |
| Batch Size | 32 |

from 4 adjacent features. If the same input size is used, this makes for a significant lower number of weights in the fully-connected layer (around 16,000) compared to the Large Model.

## 4 Evaluation

### 4.1 Data Set

The actual CLEF 2019 Bots and Gender Profiling Task test data set is hidden and only used for official system submissions. Thus, we used the training and validation data set of the CLEF 2019 Bots and Gender Profiling Task data set for training and testing our models before the CLEF submissions. In Table 1, we outline noteworthy statistics about the used data set. Note that the data set provided by CLEF is balanced, i.e., the number of bot profiles and human profiles is the same. For the official system submissions, the performance of any system is evaluated based on the accuracy. We therefore also use this evaluation metric for our own experiments.

**Table 3.** Results for the different embedding dimensions.

| Embedding | Accuracy |
|---|---|
| 100-dimensional | 84.03% |
| 300-dimensional | 85.65% |

**Table 4.** Results for the architecture variations.

| Architecture | # Trainable Weights | Train Accuracy | Validation Acc. | Precision | Recall | F1 Score | AUC Score |
|---|---|---|---|---|---|---|---|
| Large Model | 11,000,000 | 95.58% | 79.68% | 72.94% | 94.35% | 82.28% | 79.68% |
| Simple Model | 16,000 | 97.67% | 85.65% | 97.02% | 73.55% | 83.67% | 85.65% |

## 4.2 Evaluation Settings

We developed our models using Keras v2.1.2 with a Tensorflow v1.0.0 backend. Training the model was performed on a machine with 64GB memory and a GeForce GTX 1080 Ti GPU.

We implemented and evaluated our basic model using a word-based embedding method, where an embedding vector is generated for each unique word in the text corpus. The embedding formed the first layer of our model and was trained alongside the model itself. The texts are truncated to a length of 11,000 characters and we use a 300-dimensional embedding. Thus, this layer on its own adds a further 46 million parameters to be trained in the model's training process. By training the embedding as part of the model it allows the generated sequences of word indices to be the input to our model. The first layer of the network will be responsible for generating the word vectors for each of the words from the inputs.

We fine-tuned the hyperparameters of our basic model using the dedicated validation data set. In the end, we used the parameters as shown in Table 2. Note that these optimal hyperparameters performed best on both the architecture variations.

## 4.3 Evaluation Results

**Evaluating Embeddings on the Basic Architecture** Table 3 presents the evaluation results for all the used embedding dimensions. The embedding dimensions gave similar results, with some slight differences in the model accuracy in favor of the model generating 300-dimensional vectors for each word. Thus, we decided to go with that embedding in the final model for the CLEF test runs.

**Evaluating the Architecture Variations** We trained the extended models with the same hyperparameters as our basic model and used word2vec as the method of generating the word embeddings, with a dimension size of 300 based on our results from Table 3. The evaluation results for the two architecture variations are shown in Table 4.

Although the training accuracies were similar, the validation accuracies were very different. Even though the Large Model is a deeper model (having more trainable weights and predicting power), the big difference between its training and validation accuracies, led us to believe that it was overfitting on the training data. One possible reason for this might be that the model is too complex for the limited number of training samples (articles) in our task. The table also shows that the Simple Model performs better in all the other metrics as well. Thus, the model chosen for the CLEF submission was the Simple Model.

**Evaluating on CLEF's Test Data Set** Applying our basic model – using our simpler architecture variation, a word2vec embedding layer generating 300-dimensional vectors and the hyperparameters shown in Table 2 – on the official CLEF test data set via official approach submissions, we obtained an accuracy of 90.34%. Based on the accuracy alone, our model performed even better on the test data than in our experiments with the validation data. Comparing our approach to the approaches of other teams at CLEF 2019 with respect to bot identification on the English language – ignoring the other tasks proposed within the CLEF 2019 Bots and Gender Profiling Task –, our team ranked in the top half of all participating teams.

## 5 Conclusion

In this paper, we presented a convolutional neural network architecture for determining whether a Twitter feed is written by a bot or a human. In our experiments, we found that a convolutional neural network, using the flatten transition function and a 300-dimensional word2vec embedding method performs best among our methods. In the official CLEF 2019 Bots and Gender Profiling Task test runs, we obtained a relatively high accuracy of 90.34%. In the future, besides evaluating a deeper convolutional neural network, we plan to develop further approaches based on LSTMs.

## References

1. Adewole, K.S., Anuar, N.B., Kamsin, A., Varathan, K.D., Razak, S.A.: Malicious accounts: Dark of the social networks. J. Network and Computer Applications 79, 41–67 (2017)
2. Beskow, D.M., Carley, K.M.: Bot Conversations are Different: Leveraging Network Metrics for Bot Detection in Twitter. In: Proceedings of the IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining. pp. 825–832. ASONAM'18 (2018)
3. Chavoshi, N., Hamooni, H., Mueen, A.: DeBot: Twitter Bot Detection via Warped Correlation. In: Proceedings of the IEEE 16th International Conference on Data Mining. pp. 817–822. ICDM'16 (2016)
4. Daelemans, W., Kestemont, M., Manjavancas, E., Potthast, M., Rangel, F., Rosso, P., Specht, G., Stamatatos, E., Stein, B., Tschuggnall, M., Wiegmann, M., Zangerle, E.: Overview of PAN 2019: Author Profiling, Celebrity Profiling, Cross-domain Authorship Attribution and Style Change Detection. In: Crestani, F., Braschler, M., Savoy, J., Rauber, A., Müller, H., Losada, D., Heinatz, G., Cappellato, L., Ferro, N. (eds.) Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019). Springer (2019)
5. García, S., Grill, M., Stiborek, J., Zunino, A.: An empirical comparison of botnet detection methods. Computers & Security 45, 100–123 (2014)
6. Howard, P.N., Woolley, S., Calo, R.: Algorithms, bots, and political communication in the US 2016 election: The challenge of automated political communication for election law and administration. Journal of information technology & politics 15(2), 81–93 (2018)
7. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1746–1751. EMNLP'14 (2014)
8. Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. Information Sciences 467, 312 – 322 (2018)

9. Llewellyn, C., Cram, L., Favero, A., Hill, R.L.: For Whom the Bell Trolls: Troll Behaviour in the Twitter Brexit Debate. CoRR abs/1801.08754 (2018)
10. Loyola-González, O., Monroy, R., Rodríguez, J., López-Cuevas, A., Mata-Sanchez, J.I.: Contrast pattern-based classification for bot detection on twitter. IEEE Access 7, 45800–45817 (2019)
11. Lundberg, J., Nordqvist, J., Laitinen, M.: Towards a language independent Twitter bot detector. In: Proceedings of the Digital Humanities in the Nordic Countries. pp. 308–319. DHN'19 (2019)
12. Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., Tesconi, M.: Rtbust: Exploiting temporal patterns for botnet detection on twitter. CoRR abs/1902.04506 (2019)
13. Rangel, F., Rosso, P.: Overview of the 7th Author Profiling Task at PAN 2019: Bots and Gender Profiling. In: Cappellato, L., Ferro, N., Losada, D., Müller, H. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2019)
14. Varol, O., Ferrara, E., Davis, C.A., Menczer, F., Flammini, A.: Online Human-Bot Interactions: Detection, Estimation, and Characterization. In: Proceedings of the Eleventh International Conference on Web and Social Media. pp. 280–289. ICWSM'17 (2017)