Finding Bot Accounts on Twitter via Tweets

David Tomassi, Somdutta Bose, Saja Alayadhi

November 26, 2019

Abstract

The goal is to detect bot accounts on Twitter via the granularity of tweets and their contents. Recently, we have seen the phenomena of "Fake News" and people's ability to buy bot accounts to respond to tweets and make them seem more popular. The significance of the project is to be able to see which accounts on Twitter are actually human. The data set we used is the MIB Dataset (http://mib.projects.iit.cnr.it/dataset.html) which has a genuine and bot account information and their respective tweets. We tokenized tweets and did an embedding (GloVe and Word2vec) of the tokens and then appended them together to be the input to our model. We investigate the effectiveness of different approaches using a single layer feed-forward neural network, multiple layer feed-forward neural network and Long short-term memory (LSTM). The results we present, show how accurate and precise the model is at determining if a Twitter account is a bot by the tweets.

1 Introduction

Spam emails, advertisements, and communications in social media are a common part of the internet and using these services. Detecting these spam outlets has been a rich area of research [2, 3, 5, 6, 9, 10, 11, 12, 13, 17, 18]. With "Fake" news being spread throughout social media [7] it has become imperative that new methods and techniques be developed to address this phenomenon. The focus on the social media platform Twitter has become emergent [10, 11, 12, 13, 18] as bot accounts inflating the popularity of certian ideas has become common. Current approaches to detect bot accounts use hand-engineered features which can be troublesome to retrieve.

In this paper we introduce several different approaches to solve the problem of detecting a bot account on Twitter by just a tweet text. We use different deep learning approaches using a feed-forward neural network, an Long

Short-Term Memory (LSTM) based architecture, and Bidirectional Encoder Representations from Transformers (BERT) with the goal of identifying a bot account by just a tweet text. Since only a tweet is required for classification, users do not have to be encumbered with gathering a multitude of different data ranging from number of followers, age of the account, and etc. This makes the approach easier to use than previous methods.

The rest of the paper will be as follows: Section 2 we discuss some of the previous work that has been done in detection of spam Twitter accounts and in Section 3 we discuss our approach in detail. The results of our experiments are described Section 4. We have a discussion on the results in Section 5 and we conclude in Section 6. In Section 7 we detail each author's contributions.

2 Related Work

In [14], the authors propose a deep neural network based on contextual long short-term memory (LSTM) architecture. They exploit both the content and metadata to detect bots at the tweet level. They extracted contextual features from user metadata and fed as auxiliary input to LSTM deep nets processing the tweet text. They also proposed a technique based on synthetic minority oversampling to generate large dataset, suitable for deep nets training, from a minimum amount of labeled data. They demonstrated that with their architecture they achieved high classification accuracy (AUC >96%) in separating bots from humans. When they applied the same architecture to account-level bot bot detection, they achieved nearly perfect classification accuracy (AUC >99%). In [1], the authors propose a novel approach for distinguishing spam from no-spam social media post. They optimized a set of features independent of historical tweets. These tweets were available only for a short time on Twitter. Account features related to users were taken into account. They observed that an average automated spam account posted at-least 12 tweets a day at well defined periods. Their approach achieved a significant improvement on performance when compared to exiting spam detection techniques.

3 Methods

3.1 Dataset

The dataset we will be using for our training and evaluation will be the My Information Bubble (MIB) dataset [11]. It is a collection of genuine and spam tweets from bot accounts. There are thousands of accounts with millions of tweets. In one approach, we will split the dataset by accounts with a 90/10 split for training and validation data. In another approach, we will split the dataset into three sets (training, validation and testing). The split will be done by accounts level with a 80/20 for training and testing data, then a split for the training data by 80/20 for training and validation data.

3.2 Pre-Processing

We will pre-process the tweets in order to normalize their format and reduce vocabulary size when we create an embedding.

3.2.1 Parsing

To parse the tweets, we will use three different ways depending on the embedding that will be mentioned in the next section.

In the first approach, we split our tweet sentence on the basis of comma and strip the data of double quotes. We used a Word2Vec embedding [15] which decreases the dimensionality to 100 to represent one word. We tried two other approaches, since this approach did not result in good results in terms of “Accuracy”.

In one approach, we employ the text-processing tool Ekphrasis [4] that will perform tokenization, normalization and word segmentation. Using Ekphrasis allows for a uniform format for the tweets and abstracts away usernames and emojis which will decrease the vocabulary size that will be used in the embedding.

In another approach, we will preprocess the tweets based on the preprocessing script that is done by the creators of Global Vectors for Word Represention (GloVE) [16]. This will allow us to use the pre-trained word vectors for Twitter data provided by GloVE.

3.2.2 Embedding

Since our vocabulary size will be large as the breadth of language, slang, and domain specific words is diverse. Using a one-hot-encoding, would increase the dimensionality of the input by a large margin. To overcome this obstacle, we used word embedding. We can either train the word embedding in our data or use a pre trained one. In this project we tried both of these approaches. In two approaches, we used a Word2Vec embedding [15] which decreases the dimensionality to 100 to represent one word. In particular, we train a Word2Vec model[1] on our tweets and use the embedding to encode

[1]https://radimrehurek.com/gensim/models/word2vec.html

them for input to the model.

In another approach, we used the pre-trained GloVE specific in twitter data which encode the vocabularies in 100 dimension vectors [2]. In particular, we used a set of the embedding from the pre-trained GloVE to encode our tweets to fed them as input to our model.

3.3 Pruning

We pruned tweets from our dataset that fall outside the vocabularies that was learning from our Word2Vec model. With the feed-forward neural network models, we limit the length of the tokenized tweet as the input to the network has to be a fixed sized. Tweets above this length threshold have been dropped from the dataset and tweets below the threshold will be padded. While with the pre-trained Glove approach, we used zero vector to encode any vocabulary that falls outside the pre-trained Glove.

3.4 Feed-Forward Neural Network

For the first two approaches we have classified tweets in a feed-forward neural network architecture. We have tried two different implementations with a single hidden layer and multiple hidden layers.

3.4.1 Single Hidden Layer

The single layered feed-forward neural network are fully connected and have as input 20000 sized vector with a single hidden layer of 200 neurons, and an output layer with one neuron for prediction. The activation function for the hidden layer will be the Rectified Linear Unit (ReLU) function and the output layer will have the activation function as sigmoid to make the prediction.

3.4.2 Multiple Hidden Layers

The multiple hidden layered feed-forward neural network are fully connected and have as input 20000 sized vector with three hidden layers of 200 neurons each, and an output layer with one neuron for prediction. Similarly, as the single hidden layer feed-forward neural network, the activation function for the hidden layers are ReLU and the output layer is sigmoid.

[2]https://nlp.stanford.edu/projects/glove/

3.5 Long Short-Term Memory (LSTM)

In this model, we used pre-trained GloVE to encode the tweets text. So, the first part was preprocessing the tweets text to match the preprocessing for GloVE tweets, as mentioned in the parsing section. The labels were added to the dataset to recognize the ones from the genuine tweets' dataset and the spam ones. The split of the data was done following the second approach mentioned in the dataset section. We developed the model using Keras framework [8]: First layer was an embedding layer. This layer will take as an input : the maximum length of a tweet , which has been set in this approach to 140, the number of vocabularies that we have, in this approach the counting was done for the training set, and the dimension of the dense embedding, in this approach is set to 100. Since we used a pre trained Glove for the embedding, we provided this layer with the weights for the embedding and set them to be untrainable. The second layer is an LSTM layer with 100 neurons and a 20% dropout. The final, output, layer has a single neuron for prediction with sigmoid function.

3.6 Bidirectional Encoder Representations from Transformers (BERT)

BERT is available for download on Github. The input file for this tool should be in a *.tsv* file format. BERT accepts input (*train.tsv* and *dev.tsv*) in a certain format. The input format that BERT accepts is given below:

- Column 1 is the id for a row.
- Column 2 is a label for the row. This is an integer. These are classification labels that our classifier aims to predict.
- Column 3 is a column of all the same letter — this is a throw-away column that we need to include because the BERT model expects it.
- Column 4 the text examples that we want to classify.

The file *test.tsv* should be in another format. The format is given below:

- Column 1 is the id for a row.
- Column 2 is the text that we want to classify.

There are four BERT "base" models which have different pre-trained weights. We picked "cased" and "uncased". These are use for different letter cases. A script file is created to run BERT. This script file contains the location of the input data files (*.tsv*). It also contains entry to specify the location of BERT pre-trained weights.

On running this script file, we got an error. The error is “Assign requires shapes of both tensors to match.”. On searching for solution for this error, we found that this error may be caused because TensorFlow does not delete previous checkpoints. However, we have not found the correct solution to this problem.

4 Results

4.1 Feed-Forward Neural Network

Precision and recall are extremely important metrics to evaluate out model. Unfortunately, it is not possible to maximize both these scores at once. For problems for which both precision and recall are important, we can maximize the F1 score.

4.1.1 Single Hidden Layer

Since we did obtain a high “Accuracy” with manually pre-processing the data, we used the tool Ekphrais for the same. Without Ekphrasis we got an accuracy of 36.08%. With Ekphrais, we got a accuracy of 87.3% for a single hidden layer feed-forward network.

4.1.2 Multiple Hidden Layers

Without Ekphrasis we got an accuracy of 45%. With Ekphrasis we obtained and acuracy of 93.5% for multiple hidden layer feed-forward neural network.

4.2 Long Short-Term Memory

This model achieved accuracy of 83.5% on the testing set and 89% on the validation set. Results of the modification in hyper-parameters and also the model architecture with pre-trained emending that all came with worse results are mention in the discussion section.

5 Discussion

5.1 Feed-Forward Neural Network

Both feed-forward networks achieved good accuracy on the validation set. The multiple hidden layer architecure was able to perform better than its single layer counter part. This could be because of the deeper layers learning more complex relationships between the tokens in the tweet, as the single hidden layer was learning patterns of spam tweets.

5.2 Long Short-Term Memory

The assumption was that LSTM would achieve a batter accuracy, but it did not. In our problem the single hidden layer feed forward network achieved a better accuracy. However, we couldn't tell if the difference in the accuracy because of the model choice or the embedding approach since we used different approach for each model. Therefore, we tried to use the same model architecture that we used in the multiple hidden layered feed-forward neural network approach with this pre-trained GloVE embedding. The results was worse than LSTM with accuracy for testing data: 79% and in the validation data: 83% Fig. 1

One of the future work that can be done here is to do the same LSTM model with word embedding that are trained in our data instead of using the pre trained Glove to test the other hypothesis that embedding by training on our tweets is the reason of the better accuracy.
In order to try to in- crease the accuracy with the pre-trained embedding, different hyper-parameters for the model were examined but none of which achieved better result. Moreover, we modified the model architecture to include convolution layer with max pooling but the accuracy did not improve. Another suggestion for future work is to increase the number of neurons in the LSTM layer but that would need of course more computational power. Below is a summary of the different changes to the model and the results:

- Adding more neurons to the LSTM layer, to be 400:
  The result: the accuracy on testing data is 83% and accuracy on validation data is 90%.Fig. 2

- Changing the data split to have more training samples, to be 90% training data:
  The result: the accuracy on testing data is 83% and accuracy on validation data is 89%. Fig. 3

- Change the dictionary for the embedding to include only the testing data without the validation data to simulate the testing data status. The result: accuracy on testing data is 83% and accuracy on validation data is 89%. Fig. 4

6 Conclusions

This project aimed to classify a twitter account as a spam or not spam by using only a tweet text. We successfully got a good accuracy in two different approaches for the word embedding and tried to develop four models. The best accuracy obtained was 93.5%, from training using multiple hidden layer feed-forward neural network with Word2Vec embedding. We also observe that we got better results with Word2Vec embedding. However, for future work can involve LSTM with Word2Vec embedding and using GloVe with single-layer feed-forward neural network and multiple-layer feed-forward neural network.

7 Author Contributions

7.1 David Tomassi

- Requesting Access to MIB Dataset
- Word2Vec Embedding
- Single Hidden Layered Feed-Forward Neural Network
- Multiple Hidden Layered Feed-Forward Neural Network

7.2 Somdutta Bose

- Single Hidden Layered Feed-Forward Neural Network with data preprocessed with and without Ekphrasis tool.
- Multiple Hidden Layered Feed-Forward Neural Network with data preprocessed with and without Ekphrasis tool.
- Worked on BERT

7.3 Saja Alayadhi

- Pre-trained Glove embedding approach.
- The Long Short-Term Memory model approach.
- Multiple Hidden Layered Feed-Forward Neural Network with pre-trained Glove embedding.

References

[1] Alom, Z., Carminati, B., Ferrari, E.: Detecting spam accounts on twitter. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1191–1198 (Aug 2018), ISSN 2473-9928, doi:10.1109/ASONAM.2018.8508495

[2] Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C.D., Stamatopoulos, P.: Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. CoRR **cs.CL/0009009** (2000), URL `http://arxiv.org/abs/cs.CL/0009009`

[3] Androutsopoulos, I., Paliouras, G., Michelakis, E.: Learning to filter unsolicited commercial e-mail (2006)

[4] Baziotis, C., Pelekis, N., Doulkeridis, C.: Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp. 747–754, Association for Computational Linguistics, Vancouver, Canada (August 2017)

[5] Bickel, S., Scheffer, T.: Dirichlet-enhanced spam filtering based on biased samples. In: Proceedings of the 19th International Conference on Neural Information Processing Systems, pp. 161–168, NIPS'06, MIT Press, Cambridge, MA, USA (2006), URL `http://dl.acm.org/citation.cfm?id=2976456.2976477`

[6] Bratko, A., Filipič, B., Cormack, G.V., Lynam, T.R., Zupan, B.: Spam filtering using statistical data compression models. J. Mach. Learn. Res. **7**, 2673–2698 (Dec 2006), ISSN 1532-4435, URL `http://dl.acm.org/citation.cfm?id=1248547.1248644`

[7] Chiou, L., Tucker, C.: Fake news and advertising on social media: A study of the anti-vaccination movement. Working Paper 25223, National

Bureau of Economic Research (November 2018), doi:10.3386/w25223, URL `http://www.nber.org/papers/w25223`

[8] Chollet, F., et al.: Keras. `https://keras.io` (2015)

[9] Cormack, G.V.: Email spam filtering: A systematic review. Found. Trends Inf. Retr. **1**(4), 335–455 (Apr 2008), ISSN 1554-0669, doi: 10.1561/1500000006, URL `http://dx.doi.org/10.1561/1500000006`

[10] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: Fame for sale: efficient detection of fake twitter followers. Decision Support Systems **80**, 56–71 (December 2015), ISSN 0167-9236, doi: http://dx.doi.org/10.1016/j.dss.2015.09.003

[11] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In: Proceedings of the 26th International Conference on World Wide Web Companion, pp. 963–972, WWW '17 Companion, International World Wide Web Conferences Steering Committee (2017), ISBN 978-1-4503-4914-7, doi:10.1145/3041021.3055135, URL `https://doi.org/10.1145/3041021.3055135`

[12] Inuwa-Dutse, I., Liptrott, M., Korkontzelos, I.: Detection of spam-posting accounts on twitter. Neurocomputing **315**, 496 – 511 (2018), ISSN 0925-2312, doi:https://doi.org/10.1016/j.neucom.2018.07.044, URL `http://www.sciencedirect.com/science/article/pii/S0925231218308798`

[13] Krishnan, S., Chen, M.: Identifying tweets with fake news. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI), pp. 460–464 (July 2018), doi:10.1109/IRI.2018.00073

[14] Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. Information Sciences **467** (02 2018), doi:10.1016/j.ins.2018.08.019

[15] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, pp. 3111–3119, NIPS'13, Curran Associates Inc., USA (2013), URL `http://dl.acm.org/citation.cfm?id=2999792.2999959`

[16] Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on

Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, Association for Computational Linguistics, Doha, Qatar (Oct 2014), doi:10.3115/v1/D14-1162, URL https://www.aclweb.org/anthology/D14-1162

[17] Solan, E., Reshef, E.: The effects of anti-spam methods on spam mail. (01 2006)

[18] Yardi, S., Romero, D., Schoenebeck, G., danah boyd: Detecting spam in a twitter network. First Monday **15**(1) (2009), ISSN 13960466, doi: 10.5210/fm.v15i1.2793, URL https://firstmonday.org/ojs/index.php/fm/article/view/2793

8 Appendix

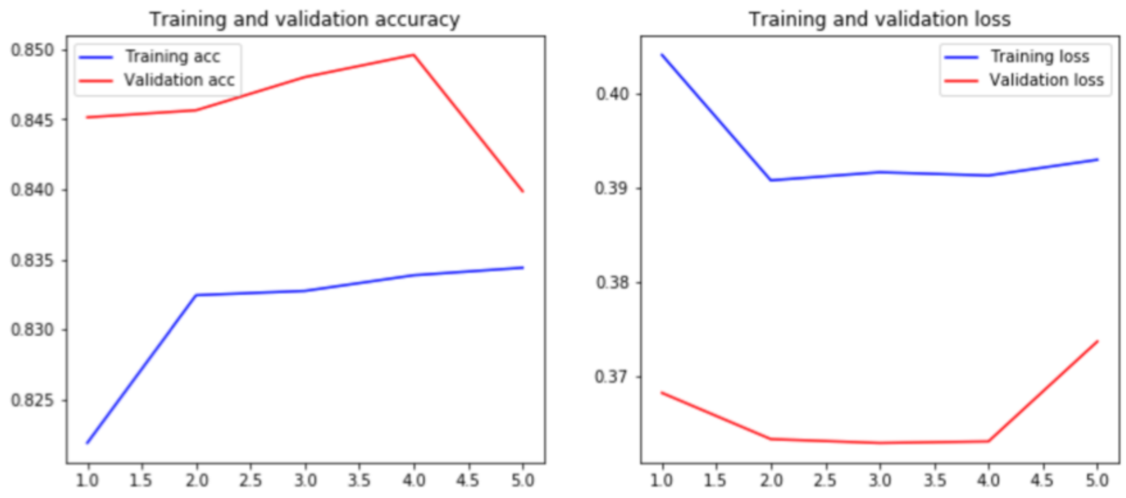

Figure 1: multiple hidden layered feed-forward neural network approach with pre-trained GloVE embedding where x-axis is the epochs and y-axis is: the accuracy on the left and the loss on the right

.

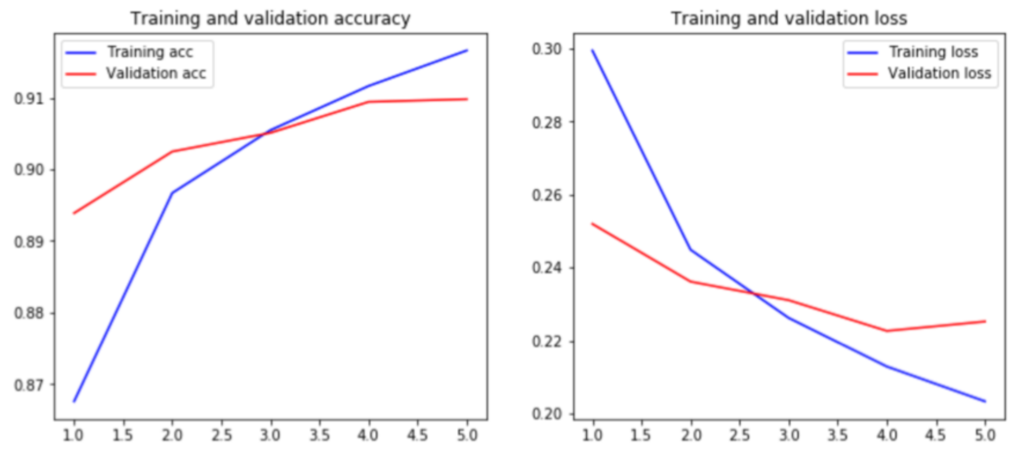

Figure 2: LSTM layer with 400 neurons where x-axis is the epochs and y-axis is: the accuracy in left and the loss on the right.

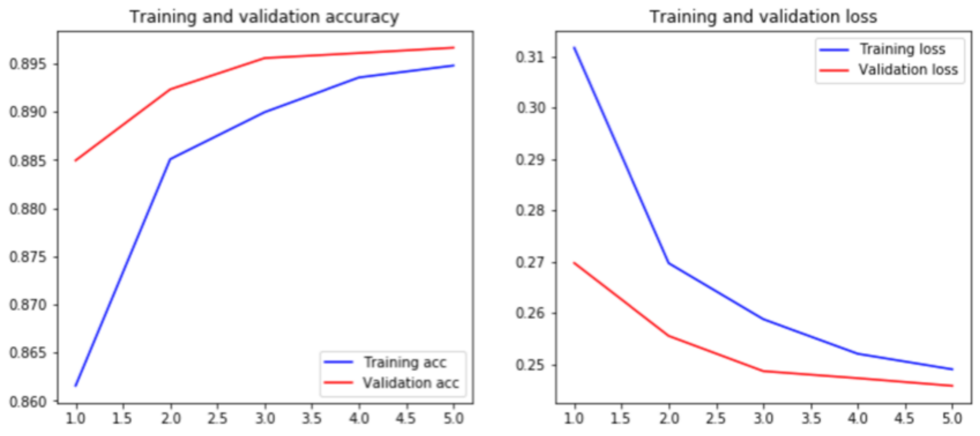

Figure 3: LSTM model with 90% for training data where x-axis is the epochs and y-axis is: the accuracy on the left and the loss on the right.

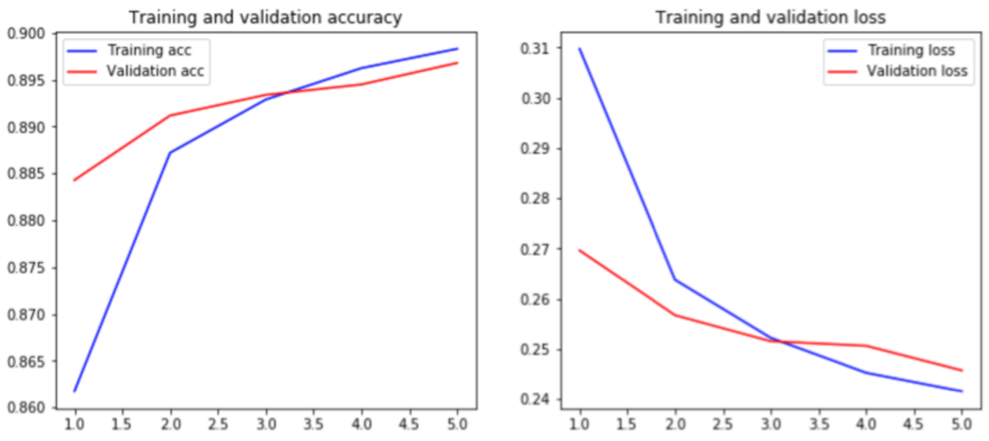

Figure 4: LSTM with embedding dictionary that doesn't have vocabularies from the validation data where x-axis is the epochs and y-axis is: the accuracy on the left and the loss on the right .