Twitter bot detection using deep learning

Yeolekar, Krishna
Computer Science,
Illinois institute of technology
Chicago, IL

Abstract— **A social bot is an autonomous bot which communicates autonomously on social media. Twitter bots can do various tasks ranging from simple tasks like retweeting to complex ones like writing tweets based on some keywords. Bot detection is important as bots can be used to spread misinformation and propaganda. Most of the techniques work by considering twitter accounts. These techniques consider user metadata to detect if a user is a bot or human. In this paper we have used machine learning and deep learning techniques to detect if an account is bot or human. In the current implementation we have used three methods - Decision tree, Naive bayes and random forest for bot detection. We have achieved a testing accuracy of 87%, which will be further increased to about 96% by using deep learning methods in the next phase. While working on this paper, we have realized that deep learning is not a one stop solution for all problems as some machine learning techniques can be used to achieve the same accuracy. Furthermore, as the project is developed further we will be adding a web api which would use our pre-trained model to classify if an account is bot or human. We have used a labelled dataset which consists of 2800 twitter accounts. In the future, we hope to increase the accuracy of the model and create a tweet-based classifier which would detect if a tweet was written by a bot or human**

Keywords—Machine learning, Twitter bots, deep learning.

I. Introduction

Social media like Twitter, Facebook and Instagram are widely used platforms which are used as interaction tools by millions of users. It is estimated that between 9% to 15% of all active twitter accounts are bots which account to almost 48 million accounts [2]. The original task of social bots was to automate simple or repetitive tasks like sending encouraging messages to users exhibiting signs of depression, automated messages showing the location and magnitude of any recorded earthquake and sports scores [3]. But malicious actors found a way to use these bots for manipulating people and spreading their propaganda. During the 2016 US elections it is estimated 400,000 bots drove nearly 20 percent of all political-related chatter the day before the election. [3]

The sophistication of bots can range from simple bots which performs actions like retweet, liking a tweet to complex bots which can interact with other users. As natural language processing becomes better the task of detecting bots will become more difficult. Thus, we need to devise new techniques which can efficiently detect bots on social media like twitter.

One of the proposed solutions to this is while account creation. The user sign-up process could be used to detect these bots and prevent their account creation altogether. However, sign-up process is actually the Achilles heel of social networking sites in this context since they are under heavy business pressure to extend their user bases. Therefore, they cannot employ complicated techniques to detect bots as they can discourage humans to sign-up. Furthermore, a user account registered by a human can later be used by a bot. [4]

II. Prior Work

Most of the previous work done was on account level bot detection. Account level bot detection refers to using account's metadata features like number of tweets, number of followers, etc to detect if the account is bot or human. Prior works like "Supervised Machine Learning Bot Detection Techniques to Identify Social Twitter Bot" used machine learning to detect bots on twitter and achieved an accuracy of 95.77% with a misclassification rate of 4.23%. This was done with a total dataset of 8,386 total accounts; 4,912 social spambots and 3,474 real accounts [5]. In this approach they have used logistic regression for preprocessing of data and then used support vector algorithm on the data. Analysis of bot accounts show many behaviors that are different from human behaviors. These behaviors can be categorized as tweet syntax, tweet semantics, temporal behavior, user profile information and user network [4]. Tweet syntax includes information about contents of the tweets like hashtags, mentions, URL's, special characters, as well as statistics such as number of retweets and location information. Basically, these features are to detect if tweet content is generated by an automated agent [4].

Another approach [9] focuses on the classification of human, bot and cyborg accounts on Twitter. Cyborgs are either bot-assisted human or human-assisted bot. The authors collected data of over 500,000 accounts and observe the difference among human, bot and cyborg in terms of tweeting behavior, tweet content, and account properties. A classification system was developed by the authors consisting of four parts (1) an entropy-based component, (2) a machine-learning-based component, (3) an account properties component, and (4) a decision maker. The system uses the combination of features extracted from an unknown user to determine the likelihood of being a human, bot or cyborg.

SybilRank[10] is another tool which was developed to detect fake accounts. It relies on social graph properties to rank users according to their perceived likelihood of being fake (Sybils). SybilRank is computationally efficient and can scale to graphs with hundreds of millions of nodes.The authors[10] deployed SybilRank in Tuenti's operation center and found that ~90% of the 200K accounts that SybilRank designated as most likely to be fake, actually warranted suspension.

III. Approach

The first task of any machine learning or deep learning task is collecting appropriate data which can be used to create a effective model. We have used a dataset[11] which has over 8,000 user accounts and over 11 million tweets. The dataset is divided into various files for real users and bots, which makes it easy to train them. As they are labelled we used the supervised machine learning approach to create a model which is efficient. Currently, we are using all features of the dataset and some of these features can be eliminated to improve the accuracy of the model. Previous works show that a minimal number of features can be used to gain maximum accuracy [1] for account level detection.

These features are -
• Statuses Count
• Followers Count
• Friends Count
• Favorites Count
• Listed Count
• Default Profile
• Geo Enables
• Profile Uses Background image
• Verified
• Protected

For tweet level detection we only used the tweets written by users. This might be improved upon and a few more features can be added to this model.

IV. Methods

4.1 Account level classification

We have used all features of the dataset with no preprocessing and used three machine learning methods to create a optimal model. We have used decision tree, Naive bayes and random forest classifier for the current implementation of the project.

4.1.1 Data Preprocessing

Convert data type to numeric - All data is converted to numerical form and attributes such as username, screen_name, etc are skipped from this conversion.
Introduce new attribute 'bot' which has value 0/1 - As the dataset has different files for genuine and fake accounts we introduce a new attribute which labels the data.

4.1.2 Feature Selection

There are 43 total features in the dataset. We select only those attributes which give any meaningful information - So we select only 10 relevant features

Selected Features :

- statuses_count
- followers_count
- friends_count
- favourites_count
- listed_count
- default_profile
- geo_enabled
- use_background_image
- protected
- Verified

4.1.3 Model Training

Initially, we split the dataset into train and test dataset. We split it into 80% training data and 20% test data. After the split we train various models to find the best model using accuracy as the decision parameter.

4.1.3.1 Decision Tree

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label [6]. This approach helped us achieve an accuracy of 87%.

4.1.3.2 Naive Bayes

Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting. It can also be represented using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification, and are a traditional solution for problems such as spam detection [7]. Unfortunately this model is inefficient in bot detection and yields an accuracy of only 67%.

4.1.3.3 Random Forest

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks [8]. We achieved an accuracy of 99% using this approach.

4.1.4 Web Portal

After successfully training the account level detection model the next feature to add was a web interface which takes input as the twitter username and gives output as bot or not bot. The working of this web portal is rather simple.
The portal is powered using python's microservice framework flask. We created a simple http web server using flask and created a form which takes a input as the twitter username. After the user submits the form using post request on a certain url, we get all the metadata from twitter's api which we save into our datastore. After we get the relevant data from twitter we pass this data as test data to our trained model. Our trained model gives an output based on its trained data whether the account is a bot or human. We

store this information and pass it on to the HTML page where the user sees the result of the trained model.

4.2 Tweet level detection

This method solves the problem of identifying whether a certain user is a bot or human based on the 140 characters of tweet they write rather than relying on the account metadata. For this detection we used state of the art NLP technique called Long Short Term Memory (LSTM) model, a superior variant of Recurrent Neural Networks. To transform the tweets into a form suitable for LSTMs, as an embedding we use a pre-trained set of Global Vectors for Word Representation (GloVE) meant for Twitter data [13]. GloVE is a global log-bilinear regression model that global matrix factorization and local context window methods to effectively learn the substructure of natural language, by training on word co-occurrence.

4.2.1 LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning[12]. Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer" (that is, it can compute anything that a Turing machine can). It can not only process single data points (such as images), but also entire sequences of data (such as speech or video)[12].

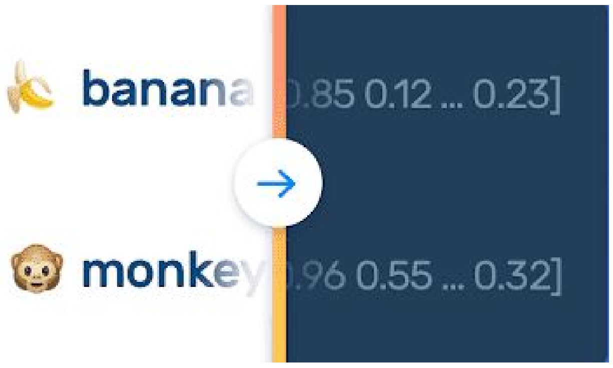


Fig : Glove Embedding

4.2.2 Data Preprocessing

Introduce new attribute ‘bot’ which has value 0/1 - As the dataset provides us with different files for genuine and fake tweets we introduce a new attribute ‘bot’.

Clean text - We clean tweet text so that there are no url’s, hashtags, etc.

4.2.3 Feature Selection

We use only one feature for training this model which is the tweet text. We pass this tweet to our mode which is labelled as either bot or not.

4.2.4 Model Training

We used keras to train our LSTM model with the data of over 11 millions tweets. With all the preprocessing done to the tweets we only used the text and omitted all other features like URL, hashtags, etc.

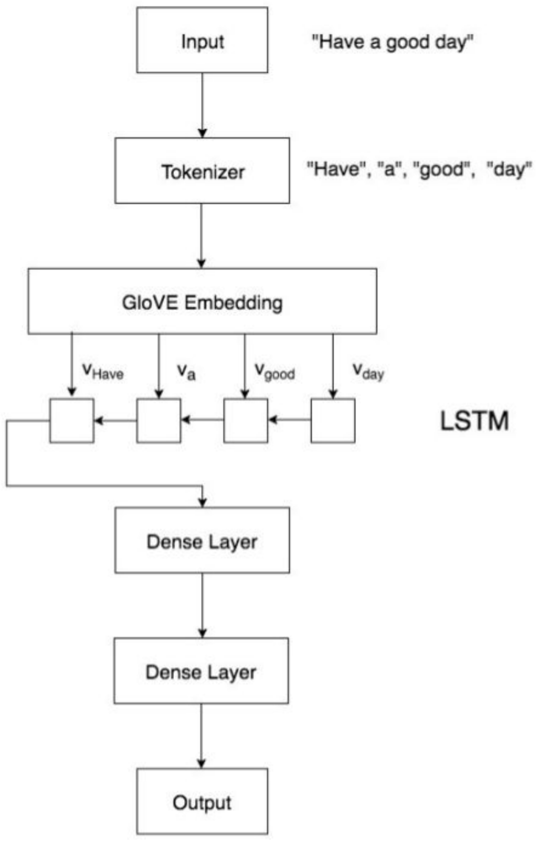


Fig : Twitter bot detection deep learning architecture

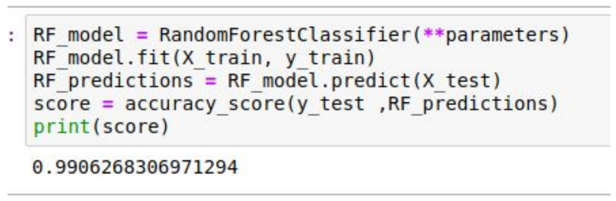
V. Results

5.1 Account level classification

As discussed before we used random forest classifier to train our model and got over 99% accuracy with this model. We achieved this accuracy without much pre-processing. We used scikit learn to implement the model and used python to implement it.

5.2 Tweet Level Classification

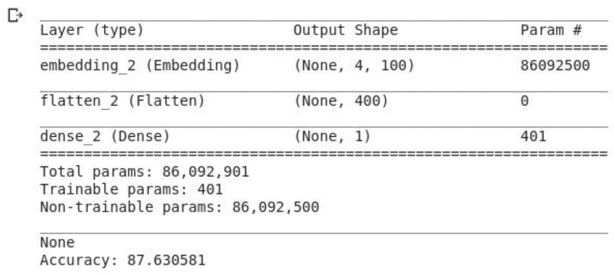
By using a simple deep learning model using keras and python. We achieved an accuracy of 87% by just passing three layers.



```
: RF_model = RandomForestClassifier(**parameters)
  RF_model.fit(X_train, y_train)
  RF_predictions = RF_model.predict(X_test)
  score = accuracy_score(y_test ,RF_predictions)
  print(score)

  0.9906268306971294
```

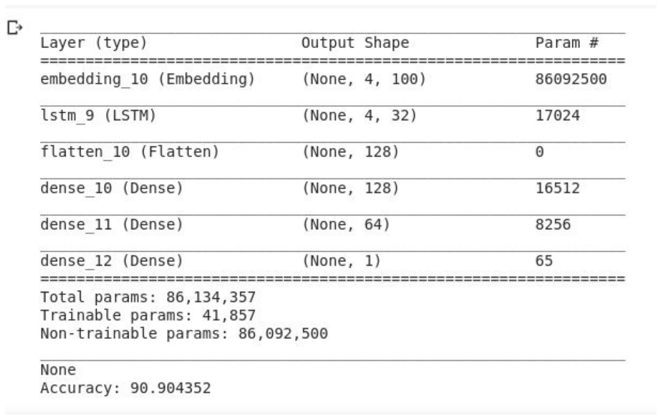
Fig : Accuracy of Account level detection.



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 4, 100)            86092500
_________________________________________________________________
flatten_2 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 401
=================================================================
Total params: 86,092,901
Trainable params: 401
Non-trainable params: 86,092,500
_________________________________________________________________
None
Accuracy: 87.630581
```

Fig : Accuracy of crude deep learning system.

We improved this model and used LSTM to increase the accuracy by 3% to 90% in the final model. We can further increase the accuracy of the model by implementing better preprocessing models which take into consideration URL's, hashtags, and other features.



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_10 (Embedding)     (None, 4, 100)            86092500
_________________________________________________________________
lstm_9 (LSTM)                (None, 4, 32)             17024
_________________________________________________________________
flatten_10 (Flatten)         (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_11 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 65
=================================================================
Total params: 86,134,357
Trainable params: 41,857
Non-trainable params: 86,092,500
_________________________________________________________________
None
Accuracy: 90.904352
```

Fig : Accuracy of LSTM deep learning model.

VI. Analysis

From analyzing the results we can find certain features in a twitter account to make the model we create more effective and efficient. There are certain user features which can be considered like:

- Is the user's profile photo from a stock database
- Does the users profile has a URL
- Does the username appear autogenerated
- Number of posts/retweets/replies/mentions
- Number of sources used like mobile application, desktop browser
- Similarity of various profiles using similarity algorithms like Jaccard similarity

VII. Conclusion and Future work

Social bots are still prevalent in today's social networks. We used two techniques to detect bots. The first technique we used was based on user accounts, and we used random forest classifier to detect bots for this technique. We achieved an accuracy of 99% using only limited features of our large training data. The second technique we used was based on tweets written by users or bots. We used deep learning to achieve an accuracy of 90%. We used a method called LSTM to achieve this accuracy.

For the future work one can write a web service for tweet level detection so that based on 140 characters of tweet the system could output if it was written by a user or a bot. We can also increase the number of features for tweet level detection method as this could improve the accuracy of the system. Another future work could be using more datasets for the model as the current dataset could introduce bias as it is a bit skewed. From the results we can see that Follow bots have the tendency of unfollowing after a couple of days after the initial follow date, so In that case we need to find more data about what the user does and when they do a particular action/activity like follow, retweet, etc.

There are certain features that can be considered while creating a improved bot detection system like

- Check if user's tweets were similar to the ones created by natural language generation program and auto generated language.
- Average number of hashtags, user mentions, links, special characters in a tweet.
- Average number of retweets by the user.
- Whether the tweets were geo-enabled.
- Percentage of tweets ending with punctuation, hashtags, or links (these may be automatically generated).
- Number of languages in which tweets were generated (accounts posting tweets in many languages may be bots)
- Sentiment inconsistency.

References

[1] Deep Neural Networks for Bot Detection

[2] https://www.cnbc.com/2017/03/10/nearly-48-million-twitter-accounts-could-be-bots-says-study.html

[3] https://zignallabs.com/what-is-a-social-media-bot/

[4] https://ieeexplore.ieee.org/document/8093483

[5] https://scholar.smu.edu/cgi/viewcontent.cgi?article=1019&context=datasciencereview

[6] https://www.geeksforgeeks.org/decision-tree/

[7] https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54

[8] https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

[9] Who is Tweeting on Twitter: Human, Bot, or Cyborg?

[10] Aiding the Detection of Fake Accounts in Large Scale Social Online Services

[11] The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race

[12] https://en.wikipedia.org/wiki/Long_short-term_memory