Use Machine Learning Methods to Detect Twitter Bots

Matthew Adams[1], Jingfeng Liu[2], Zhijian Ren[2]
[1]Department of Chemical Engineering,Carnegie Mellon University, Pittsburgh, USA
[2]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, USA

Abstract

In our project, multiple classification algorithms are implemented to distinguish Twitter bots from human accounts. These algorithms, namely logistic regression, KNN, MLP, SVM, Random Forest and XGBoost, are tested on continuous features including account age, number of followers, etc. We found out tree-based classification algorithms have the best performance on this detection problem compared with others. Moreover, we also use different methods to deal with the imbalanced dataset to improve classification accuracy on Twitter bots. Finally, we introduce binary features and implement discretization support vector machine (D-SVM) to increase detection accuracy with more information.

1. Introduction

Social media has been becoming more and more popular during the last few years. People find it more convenient to share their moments and chat with their friends. Sometimes we are surprised to see that their posts are liked or commented on by other we are - unfamiliar with. Some of these users are amazingly not human. Instead, they are created by some programs to perform certain tasks in the social media which we call them bots. While some Twitter bots might be useful, many are programmed to spread misinformation through mass tweeting and retweeting of falsified statements. With enough bots spreading the same message, this information will often appear in the twitter feed of actual people, which often leads to real people believing and spreading the false information they acquired from bots. Although Twitter is actively trying to deal with these bots, the society is still plagued by this issue today.

Emilio [1] concluded three features to judge whether an account may be a bot: i) “Periodic and regular timing” of tweets; ii) Whether the tweet content contains known spam; iii) The ratio of tweets from mobile versus desktop, as compared to an average human Twitter user. Although we are not able to accurately classify which information are known spam, we still can detect Twitter bots based on their account

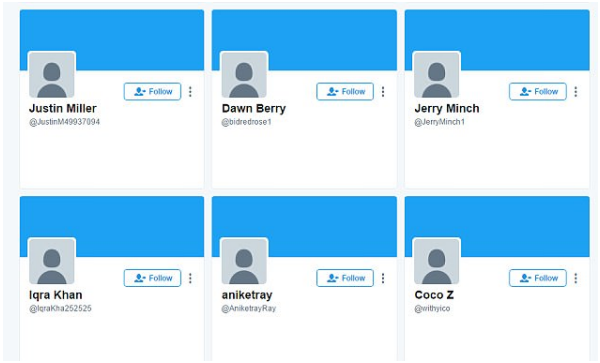

Figure 1. Some examples of active Twitter bots

profile and activities including but not limited to tweets, follows, likes and comments. While there have been a multitude of algorithms developed solely for Twitter bot detection, these bots are constantly evolving to be able to avoid detection which leads to the need for development of new methods to keep up with the bots.

The ”Bot repository” [1] is a centralized place to share annotated datasets of Twitter bots. Some of them are manually labelled for research purpose while others are purchased from companies. Since Twitter itself has been making effort to cancel many bot accounts, we use Twitter API to check if the accounts in the dataset are still active and make sure that all inactive accounts are excluded in our training and test.

We implemented several classification methods on the bot detection problem. After comparing the performance of these algorithms, we find out that tree-based algorithms are better than the others. More effort were applied on dealing with unbalanced dataset and introducing binary features.

2. Related Work

As the presence of bots on social media platforms has been rapidly increasing, there has been a constant war between bots and their detectors. Due to this, new ways of bot detection are constantly being developed as a means to

[1]https://botometer.iuni.iu.edu/bot-repository/index.html

prevent the spread of misinformation caused by these bots. These past detectors had implemented more complex feature analysis methods which involved text analysis each individual tweet that the account posted. In addition to this, previous studies also had access to other "discontinued" objects such as account language and time setting. It is possible that these features are reserved for partners that directly work with Twitter in improving bot detection.

Due to lack of access to previously supported user data objects and a rate limitation on the amount of data we can retrieve from the Twitter API[2], we used all the possible features that were able to be numerically quantified. In addition, we used a similar approach to a previous study in using a binary matrix, but our method still lacked text analysis. While lacking text analysis, our datasets's advantage is its simplicity which allows for faster data collection. If we are able to obtain similar accuracy in bot detection while remaining relatively simple, our simple dataset will allow for the analysis of large quantities of accounts in a much shorter time frame.

3. Data

In order to train our algorithms, we used published data sets which contained twitter user IDs which were labeled either as a human or bot account. Two different data sets were randomly chosen to be used to train and test our algorithms[2, 3]. To access individual user information from Twitter, we used the Python library Tweepy, which provides access to the twitter API. With access to the API and repository data, individual account features supported by the Twitter API were able to be easily processed into a readable dataframe for our machine-learning algorithms. The Twitter Developer website contains a User Data Dictionary[3] which contains a list of user characteristics that are currently supported by the Twitter API.

Our choice of account features was limited due to both time restraints and lack of access to specific account features such as account geolocation and timezone. Twitter has a set limit on the rate which one can pull data from the Twitter API , meaning that this study was limited to smaller data sets. Additional features that have proven to be helpful in bot detection such as the average number of favorites or retweets for individual user tweets would require a longer timescale as a single account can have thousands of tweets. Given these restraints, We chose the available continuous features based on their significance in previous studies.

As seen in Figure 2, the chosen continuous features show how the bots account for a majority of interactions despite having few followers when compared to genuine human accounts. Our data of continuous features have different

[2]https://developer.twitter.com/en/docs/basics/rate-limits

[3]https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object

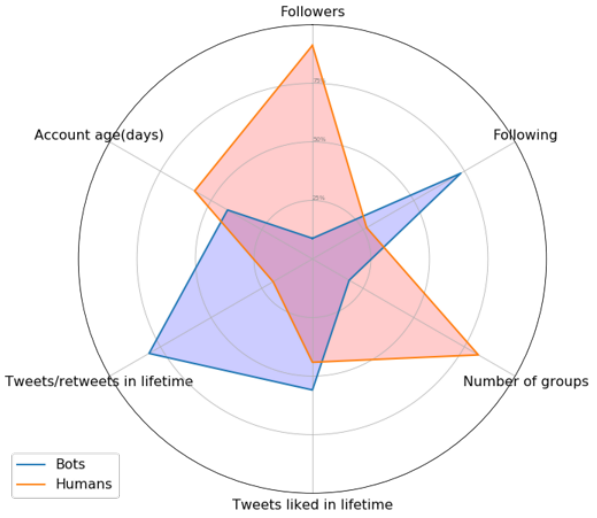

Figure 2. Comparison of continuous features between humans and bots

Default Profile Image
Default Profile Banner
Default Profile Description
Description contains a link
50:1 Following/Follower Ratio
100:1 Following/Follower Ratio
Less than 50 Followers
No Location Listed
Less than 50 Tweets
More than 1000 Tweets

Figure 3. List of Binary Features

scales. Taking that into consideration, we normalize our data so that $\mu_{data} = 0$ and $\sigma_{data} = 1$.

Since the binary features were unable to be converted to continuous, the continuous features were converted to binary form through the use of certain thresholds such as number of followers, following:follower ratio, etc.

The binary matrix consists of 10 features based on conditional statements where True = 1 and False = 0.

4. Methods

Since we have both continuous and non-continuous (binary) features, we explore classification algorithms for both of them.

4.1. Continuous Features

4.1.1 Logistic Regression

For i_{th} data point X_i, we have:

$$Pr(Y_i = 1|X_i) = \frac{exp(\beta X_i)}{1 + exp(\beta X_i)}$$

Our classification result $\hat{Y_i}$ will be:

$$\hat{Y_i} = 1, Pr(Y_i = 1|X_i) >= 0.5$$

$$\hat{Y_i} = 0, Pr(Y_i = 1|X_i) < 0.5$$

In our method, we also add L1 norm to avoid overfitting.

4.1.2 KNN

In our method, L2 distance is used when selecting nearest neighbors. For each data point in the test set, we select 5 nearest neighbors and assign the label with major vote policy.

4.1.3 Multi-layer Perceptron

We design a fully-connected multi-layer perceptron to solve the classification task. In the hidden layers, we have $6, 10, 20, 10, 5, 2$ hidden units respectively. For the first 2 layers, we also include batch normalization layers to achieve better performance. This network is trained with SGD optimizer, whose learning rate is $1e^{-3}$.

4.1.4 Linear SVM

In our project, we use a SVM with linear kernel to get the decision boundary. The penalty term we use is L2.

4.1.5 SVM with RBF kernel

RBF kernel or Radial Basis Function kernel provides more flexibility for the SVM decision boundary. A RBF kernel can be formulated as:

$$K(x^i, y^i) = exp(-\gamma \|x^i - y^i\|^2), \gamma > 0$$

4.1.6 Random Forest

A Random Forest is built as an ensemble of Decision Trees to perform different regression. It is trained via the bagging method, which consists of randomly sampling subsets of the training data, fitting a model to these smaller data sets, and ensembling the predictions.

In our method, we have 200 trees in the set and their maximum depth is 3.

4.1.7 XGBoost

XGBoost or Gradient boosting tree, like random forest, is a set of decision trees. The two main differences are:

1. How trees are built: random forests builds each tree independently while gradient boosting builds one tree at a time.

2. Combining results: random forests combine results at the end of the process while gradient boosting combines results along the way.

Like random forest, we have 200 trees in our set and the learning rate for gradient descent is $1e^{-4}$.

4.2. Binary Features with discretization based support vector machine (D-SVM)

Discrete values have important roles in data mining and knowledge discovery. Some researchers [4, 5] extended the boundaries of discretization to machine learning techniques such as support vector machine and obtained good results. The main idea of D-SVM is data discretization and applying discretize data into SVM.

There are many advantages of using discrete values over continuous one. Discrete features are closer to knowledge level representation than continuous ones. Data is reduced and simplified using discretization. As reported in a study [6], discretization makes learning more accurate and faster. In general, obtained results using discrete features are usually more compact, shorter and more accurate than using continuous ones; hence the results can be more closely examined, compared, used and reused.

5. Experiments

5.1. Overall Performance

We first compared the detection accuracy between several classification algorithms. Overall, all algorithms were able to successfully detect whether a Twitter account is a person's account or a bot with an accuracy of at least 70%. As seen in Table 1, decision-tree-based algorithms showed decent performance and took much less training time than the neural networks. The two datasets we used here are botometer feedback [2] and gilani [3]. The former is a relatively small dataset with most recent active Twitter accounts while the latter is a much larger and more balanced dataset. Both datasets were divided into training sets and test sets with account ratio of 3:1.

We recorded the confusion matrix and calculated F1 score for both datasets, as shown in Table 2. We found out that almost all algorithms are better at detecting human accounts than detecting bots. We assumed that the imbalanced dataset is the main reason for bot detection failure. To resolve this issue, we made some improvements on the Random Forest algorithm to solve this problem by changing the classification threshold, dropping some human account data, and adding more bot data. It is not surprising that the bot detection accuracy increased. However, the overall accuracy kept same or decreased.

Algorithm	Accuracy	
Logistic Regression	79.49%	77.78%
KNN	78.63%	74.33%
SVM	78.63%	78.4%
SVM with RBF kernel	73.5%	77.31%
MLP	82.05%	77.62%
Random Forest	84.62%	78.25%
XGBoost	84.62%	76.37%

Table 1. Overall performance on Twitter account classification. Left: botometer feedback dataset. Right: gilani dataset.

Algorithm	F1 Score	
Logistic Regression	0.87	0.71
KNN	0.86	0.68
SVM	0.86	0.69
SVM with RBF kernel	0.87	0.69
MLP	0.86	0.70
Random Forest	0.87	0.69
XGBoost	0.87	0.70

Table 2. F1 score for both datasets' classification.

5.2. D-SVM performance

The dataset has some binary features such whether the profile image is default. In order to apply these features into the detection, we discretized data during the pre-processing step. After that, we implemented support vector machine algorithm on these processed data. From Table 3, we noticed that simple binarization was good enough to improve bot detection accuracy even with an imbalanced dataset. Moreover, proper discretization resulted in a much better classification accuracy.

The parameter of SVM decision function, for example, capacity or model complexity was not affected by binarization or discretization as these process work on the dataset rather than the model. Discretization yielded the reduction in unique tuples by assigning the discretized value of the attribute to the objects whose numeric value lies in the corresponding discrete interval. Thus, we could observe that there had been a reduction in the number of support vectors per class during classification of the discretized dataset.

Algorithm	Overall accuracy	Bot detection accuracy
SVM	78.63%	24.24%
B-SVM	63.25 %	54.54%
D-SVM	87.32%	69.54%

Table 3. Comparison of SVM classification on different types of pre-processed data.

5.3. Comparison with Previous Studies

Due to the lack of features we had access to, our accuracy was significantly lower than that of previously published studies. We compared our average accuracy of our random forest algorithm with that of two different detectors. Botometer is also based on random forest classifiers, but it uses more complex metadata such as language and device usage, both of which take longer to process. The other detector by Efthimion et al. uses a matrix of binary features which inspired us to try using D-SVM with our own binary matrix. Unfortunately, our accuracy for D-SVM was still a bit lower, likely due to lack of implementation of any text analysis.

Detector	Average Accuracy
Our Project's RandForest	81.44%
Botometer v3	88.22%
Efthimion et al.[7]	96.25%

Table 4. Accuracy Comparison with Past Detectors

6. Conclusion

We manage to use different classification methods to detect Twitter bots. Among the classification algorithms, tree-based algorithms like random forest and XGBoost can achieve the best classification performance. Though the overall performance was good, the bot detection did not perform well enough. We think this is caused by unbalanced dataset. By changing the classification threshold or artificially synthesize bot data points, we are able to improve the bot detection accuracy. Finally, binary features were included by using D-SVM.

In order to improve the performance of our algorithms, more complex features are needed to further to create better distinctions between the human and bot accounts. Further details on the user's activity such as the language, syntax and times of tweets, would provide further distinction between each user. In order to implement this into our project, we would either need more time or for the rate limitation to be removed by Twitter. Additionally, access to the discontinued user traits would provide more features which would potentially improve our model's accuracy to that of previous studies.

Acknowledgement

We are thankful to Professor Amir Barati Farimani for his detailed explanation on different machine learning methods in class and to Weikun Zhen for his useful suggestions during milestone interview.

Supplemental Material

`https://github.com/adrianliuj/24784-Project`

References

[1] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[2] K.-C. Yang, O. Varol, C. A. Davis, E. Ferrara, A. Flammini, and F. Menczer, "Arming the public with artificial intelligence to counter social bots," *Human Behavior and Emerging Technologies*, vol. 1, no. 1, pp. 48–61, 2019.

[3] G. Zafar, R. Farahbakhsh, L. Tyson, Gareth Wang, and J. Crowcroft, "Of bots and humans (on twitter)," *In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 349–354, 2017.

[4] A. Bharadwaj and S. Minz, "Discretization based support vector machines for classification," 2009.

[5] A. Bharadwaj, S. Dahiya, and R. Jain, "Discretization based support vector machine (d-svm) for classification of agricultural datasets," *International Journal of Computer Applications*, vol. 40, no. 1, pp. 8–12, 2012.

[6] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Machine Learning Proceedings 1995*, pp. 194–202, Elsevier, 1995.

[7] P. Efthimion, S. Payne, and N. Proferes, "Supervised machine learning bot detection techniques," *SMU Data Science Review*, vol. 1.