Twitter Bot Detection

CSCI 4962 Final Project Report

Yifan(Elio) Li, liy57@rpi.edu

1.0 Introduction

During the past decade, social medias like Twitter and Facebook have spread to the entire world, these platforms have allowed people to express their opinions. However, due to many reports of social media manipulation such as political propaganda, scam and promotion, concerns about their abuse are rising.

The goal for this project is to identify Twitter bots using varies of techniques including logistic regression, support vector machines, oversampling, undersampling, NLP and deep neural networks.
Some of the techniques are learned and referenced from these research papers:

A. '*Supervised Machine Learning Bot Detection Techniques to Identify Social Twitter Bots*' by Phillip G. Efthimion, Scott Payne, and Nick Proferes. (*Link*)
B. '*Deep neural networks for bot detection*' by Sneha Kudugunta and Emilio Ferrara. (*Link*)

My implementation can be found here: *https://github.com/elio-li/CSCI-4962/blob/main/project.ipynb*

1.1 Dataset

The dataset comes from Botometer (*https://botometer.osome.iu.edu/bot-repository/datasets.html*), a dataset repository especially for bots. The specific dataset used on this project is 'cresci-2017', which contains 4 major categories: genuine accounts, social spambots, traditional spambots and fake followers.

Table 1-1 Dataset Components

Group Name	Description	Account #	Tweets #
Genuine Accounts	Verified Human Accounts	3,474	8,377,522
Social Spambots #1	Italian Political Bot	991	1,610,176
Social Spambots #2	Mobile App Promotion	3,457	428,542
Social Spambots #3	Amazon Product Promotion	464	1,418,626
Traditional Spambots #2	Scam URL	100	74,957
Traditional Spambots #3	Spam Job Offers	433	5,794,931
Traditional Spambots #4	Spam Job Offers	1,128	133,311
Fake Followers	Accounts Inflating the Follower # on Another Account	3,351	196,027

1.2 Statistics

Before building up the model we need to look at some of the statistics first, these data will definitely help us to choose and build our features.
Figure 1 shows the average number of followers in each account types. From the data we can see that job offers have the highest number of followers where mobile app promotions and fake followers account have the least followers. So accounts that have less than 50 followers and greater than 7000 can be considered as a potential bot.

Figure 1. Distribution of Average Followers in Different Account Types

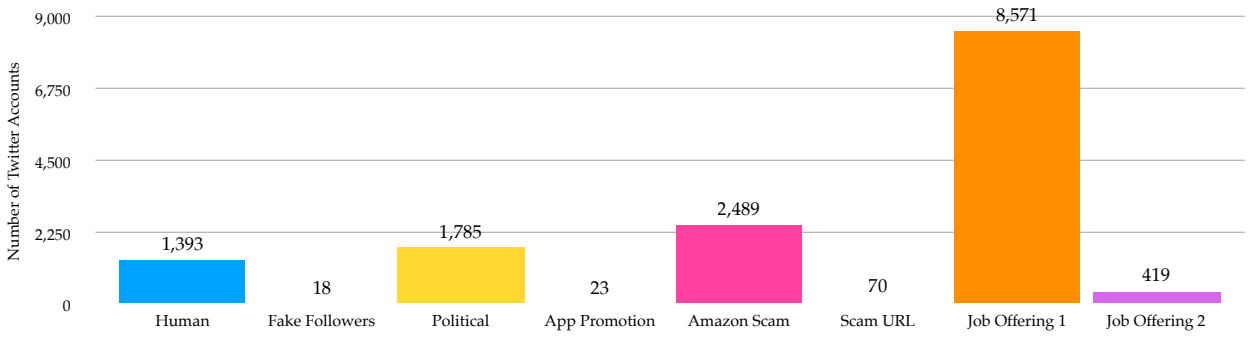


Figure 2 shows the distribution of friends in each account types, both political bots and amazon promotion bots have the highest number of friends where mobile app promotion bots & scam URL bots have the least number of friends. So we can choose a range of < 50 and > 1000 will be classified as a bot as a feature.

Figure 2. Distribution of Average Friends in Different Account Types

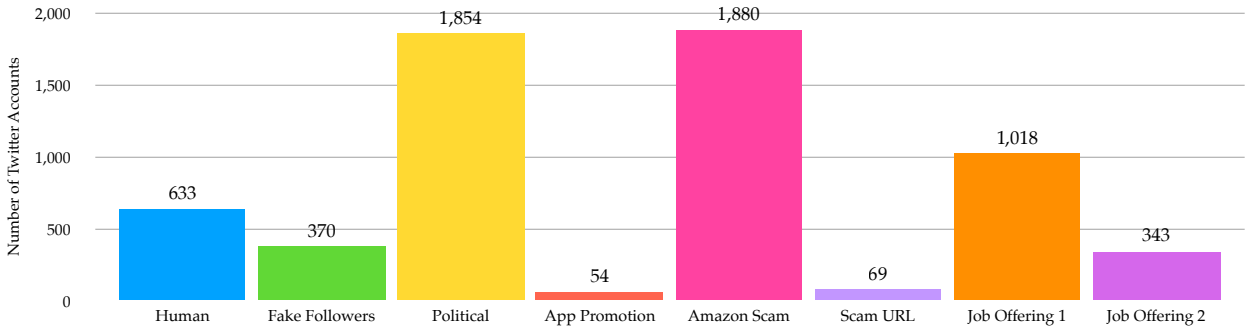


Figure 3 displays the average number of tweets per account types, amazon and job offering bots spams the largest amount of tweets, where rest of the bots spam the least amount of tweets. We can see that most of the bots account barely post any tweets, thus having a condition that checks for less number of tweets can be a good factor.

Figure 3. Distribution of Average Tweets in Different Account Types

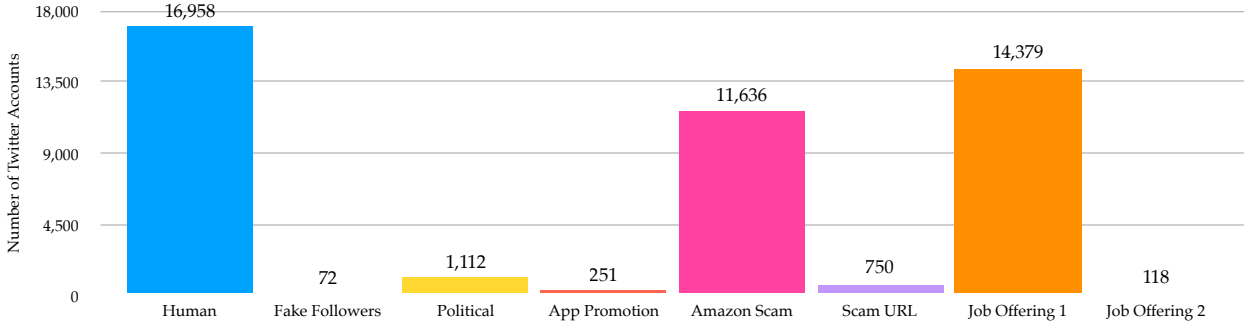


Figure 4. Distribution of Average Favorites in Different Account Types

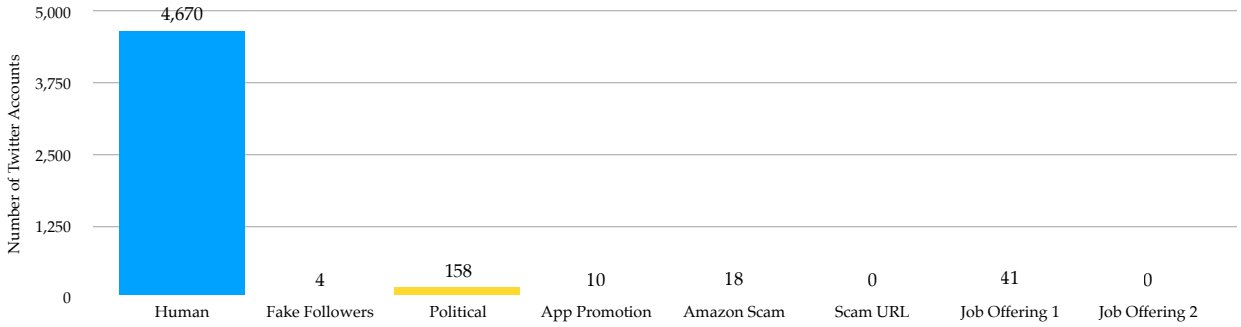


Figure 4 displays the average 'favorites' of each account types, it make sense that bots do not 'favorite' other tweets since they don't need to. This can be one of our key feature.

1.3 Features & Techniques Selection

In this project, we will classify Twitter account from 2 directions: (i) Account-Level Classification and (ii) Tweets-Level Classification.

Account-level classification will utilize the user's account information such as profile icon, whether its verified or not, number of friends, followers and tweets, etc. These features do not related to the contents that a user have tweeted, but to focus on their meta informations and determine the likelihood of being a bot. Techniques such as logistic regression, SMOTE & ENN, support vector machines and feature scaling are applied for this section.

Tweets-level classification will use user's tweets to identify whether its author is a bot or not. The approach for this section is to use Long Short Term Memory (LSTM) models. To transform the tweets text into an amenable form by using a pre-trained set called GloVe.

2.0 Account-Level Classification

As mentioned in 1.3, we will use the user's metadata as the feature. A full list of features in this section is described in Table 2-1 next page. For this classification we will use the scoring method. Each user that have satisfied the conditions for each features described in Table 2-1 will be scored 1 and 0 otherwise.

One thing I've noticed is that the classes in the dataset is completed out of balanced, where there are 9924 bot accounts and only 3474 human accounts. This imbalanced dataset might lower the accuracy of the final result and in fact, the accuracy and all other matrices have lowered for 3~5% without dealing with this issue. First we need to apply oversampling method to generate additional datas for human accounts. The methodology I picked is Synthetic Minority Oversampling Technique (SMOTE). One of the advantage of SMOTE in terms other traditional oversampling method is that it won't generate duplicates, but rather creating synthetic data points that are slightly different from the original data. Undersampling will also performed, Edited Nearest Neighbors (ENN) is selected for this case, which will remove the noisy examples that are close to each other. After performing those 2 techniques I have received 9059 bots and 7277 humans which is very balanced.

Since we have the data points, the next step will be training. The model I've chosen is Support Vector Machines. But before running SVM, I'd like to calculate and scale the weights for each feature. Since SVM is not a linear model and incapable to calculate weight. I'll apply logistic regression first, the data points are split into 3 iterations and K-Fold cross validation is applied. The accuracies for the 3 iterations are

0.99449, 0.99504 and 0.99559. Confusion matrices are described in Table 2-2. From the matrices we can see most of the errors are gathered at false positive part, this is difficult to address because there are social

Table 2-1 Account-Level Feature Descriptions

Feature	Feature Name	Function Name
Have Less than 30 Followers	`Follower`	`ScoreFollower()`
Have > 800 and < 50 Friends	`Friends`	`ScoreFriends()`
Have Favorites < 30 Tweets	`Favorites`	`ScoringFavorites()`
Have > 50 and < 10 Listed	`Listed`	`ScoringListed()`
Have < 1000 Tweets	`LessTweets`	`ScoreLessTweets()`
Have Never Tweeted	`EmptyTweets`	`ScoreEmptyTweets()`
Whether Banner Contains URL	`BannerURL`	`ScoreBannerURL()`
Whether Image Contains URL	`ImageURL`	`ScoreImageURL()`
Whether Uses Default Avatar	`Avatar`	`ScoreAvatar()`
Whether Have Description/Bio	`Description`	`ScoreDescription()`
Whether Enabled Geographic Location	`Location`	`ScoreLocation()`
Whether Account Is Verified	`Verified`	`ScoreVerified()`
Whether Account Is Protected	`Protected`	`ScoreProtected()`

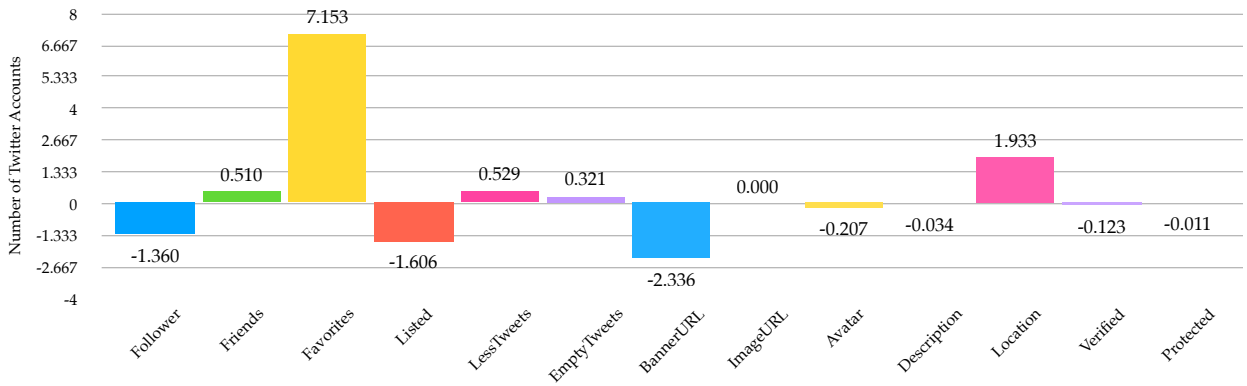
Table 2-2 Confusion Matrices for Logistic Regression

Iteration	True Positive	True Negative	False Positive	False Negative
1	2395	3021	30	0
2	2405	3013	27	0
3	2397	3024	23	1

bots that are well covered and resembling human users. However false negatives are close to 0 which is really good, if this model is going to be deployed then real human users won't get banned by mistake.

The results from logistic regression is really good but that's not essential, what is important is the weights it generated for each feature. Weights are labeled and described in Figure 5:

Figure 6. Weights for Each Features After Scaling



As we can see ‘Favorites’ feature took the largest weight as it's the most influential feature in the data points like we assumed in the statistics section. There are also some features that are less important and they have roughly 0 weights such as ‘ImageURL’, ‘Description’ and ‘Protected’. Since these features can’t impact our results as much as others do, keeping them here may also resulting overfitting, they will be removed for the support vector machines.

Lastly we will apply the scaled and normalized version of the data points into SVM, the classification result and the confusion matrix is described in Table 2-3.

Table 2-3 Support Vector Machine Reports

	Precision	Recall	F1	Support		Actual Positive	Actual Negative
0	1.0	1.0	1.0	2420	**Predicted Positive**	2412	8
1	1.0	1.0	1.0	3025	**Predicted Negative**	2	3023
Accuracy			1.0	5445			
Macro Accuracy	1.0	1.0	1.0	5445			
Weighted Accuracy	1.0	1.0	1.0	5445			

From the table we can see the result have improved in terms of logistic regression performed previously. False positive is greatly reduced and the overall accuracy, precision, recall and f1 score have all reached to approximately 100%.

We will conclude account-level classification here and next we’ll talk about tweets-level classification.

3.0 Tweets-Level Classification

For this section we will be using user’s tweets for training. Most of the state-of-the-art techniques for processing textual data in natural language processing tends to focus on inferring content style which are ineffective against advanced social bots. Thus we will be using Long Short Term Memory (LSTM) models to overcome this issue.

In order to transform the text into an amenable format fro LSTM to process, we will use a pre-trained word embedding called Global Vectors for Word Representation (GloVe) specific for Twitter data. GloVe is a log-bilinear regression model focuses on words co-occurrences over the entire corpus.

We will begin by preprocessing the data, that includes replacing occurrences of hashtags, links, numbers and mentions (@someone) into tags such as ‘<hashtag>’, ‘<url>’, ‘<number>’ and ‘<user>.’ All text will be converted to lower case and stop words will be removed.

The tokenized tweets will be transformed using GloVe model, and then will feed into LSTM that contains 2 LSTM layers with ReLU activated function and size of 128 and 64. The batch size will be 128 with an iterations of 10. The final output will be a single 32 dimensional vector.

The result is in Table 3-1, where the final accuracy is 90%, which is worse than the previous approach.

Table 3-1 Accuracy and Loss for LSTM

Iteration	1	2	3	4	5	6	7	8	9	10
Loss	0.8486	0.8795	0.8929	0.8966	0.8987	0.9003	0.9018	0.9028	0.9035	0.9037
Accuracy	0.3388	0.2835	0.2575	0.2487	0.2440	0.2403	0.2370	0.2348	0.2331	0.2330