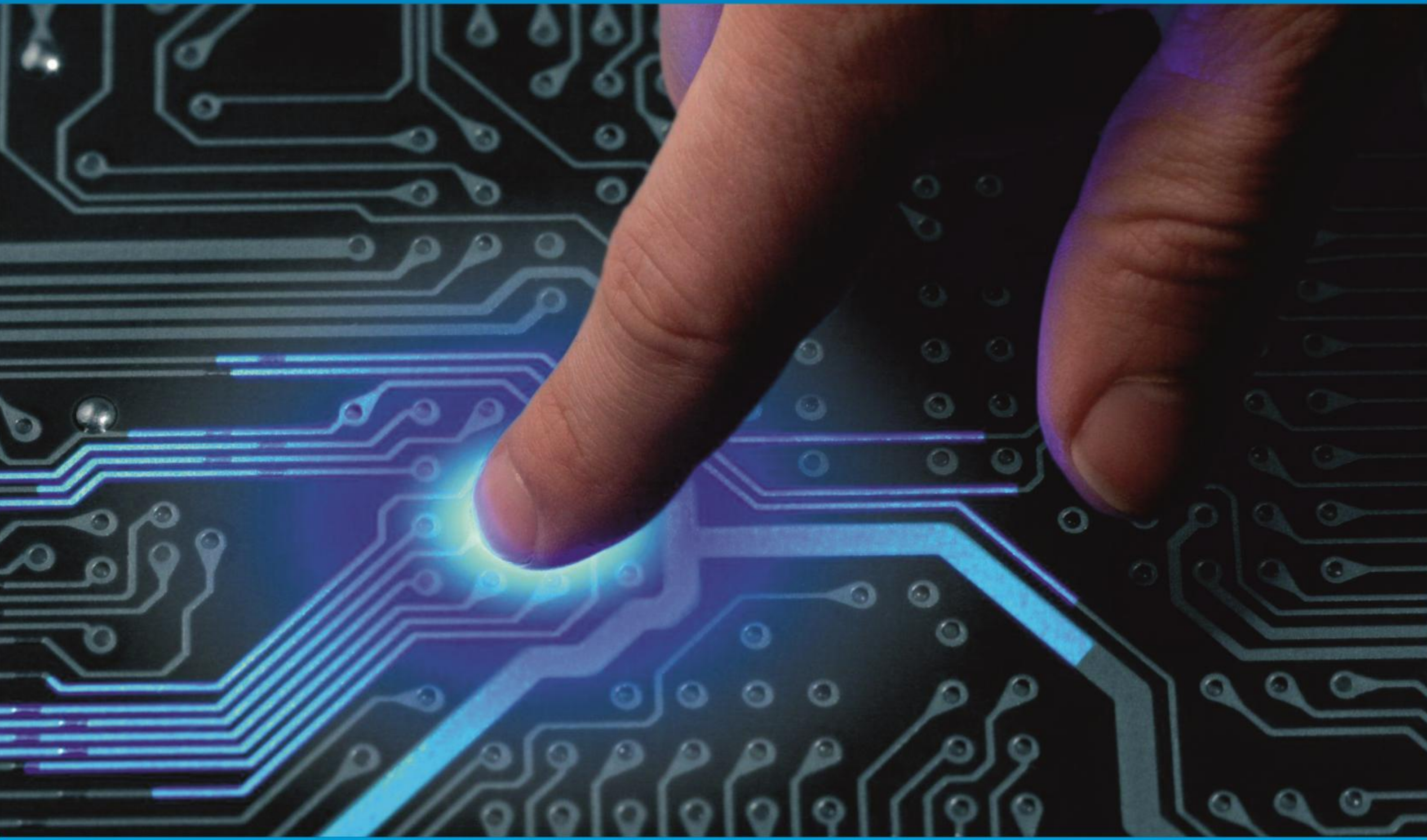
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 8, Issue 11, November 2020

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.488

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.488 |

|| Volume 8, Issue 11, November 2020 ||

Improving the Accuracy of Bot Account Detection Using Ensemble Method

Geetanjali Kushwaha, Dr. Nitesh Dubey

PG Student, Dept. of CSE., GNCSGOI Jabalpur, RGPV University, Bhopal, India

Assistant Professor, Dept. of CSE., GNCSGOI Jabalpur, RGPV University, Bhopal, India

ABSTRACT: Twitter is a popular online social network with hundreds of millions of users, where n important part of the accounts in this social network is not humans. Approximately 48 million Twitter accounts are managed by automated programs called bots, which represents up to 15% of all accounts. Some bots have good purposes, such as automatically posting information about news and academic papers, and even to provide help during emergencies. Nevertheless, Twitter bots have also been used for malicious purposes, such as distributing malware or influencing the perception of the public about a topic. There are existing mechanisms that allow detecting bots on Twitter automatically; however, these mechanisms rely on examples of existing bots to discern them from legitimate accounts. As the bot landscape changes, with the bot creators using more sophisticated methods to avoid detection, new mechanisms for discerning between legitimate and bot accounts are needed.
In this paper, we propose to use Ensemble classifier with feature engineering to enhance Twitter bot detection, as this allows detecting novel bot accounts, and requires only from examples of legitimate accounts.

KEYWORDS: Twitter bot detection; supervised classification; One-class classifiers; Anomaly detection; Social Networks, Random Forest.

I. INTRODUCTION

Detection of bots is paramount for the Twitter platform to suspend accounts that infringe on the terms and conditions. This detection uses different mechanisms, such as, accepting reports from users or flagging accounts that show suspicious behaviors such as posting the same tweet content while mentioning different users. However, bots are getting more sophisticated to achieve their goals better and avoid detection, being programmed to interact with others and mimic behaviors shown by legitimate accounts [1]. Furthermore, third parties can also benefit from bot detection, as it allows identifying and measuring the engagement an account has with real persons, or even detect command and control channels for malware botnets [2].
Given the volume of accounts and tweets, it is necessary to have automated methods for detecting bots, as even legitimate accounts can exhibit bot behaviors sometimes, making it difficult for a human to discern between them. Automated bot detection works under the premise that the behavior expressed by the account of a human differs from one of a bot. These differences can be measured using representative features such as the statistical distribution of the words used in tweets, posting rate per day, number of followers, among others. Using the extracted features, multi-class classifiers have been trained with examples of legitimate and bot accounts for classifying suspicious accounts. Most of the research in Twitter bot detection focuses on creating new feature representations and applying novel classifiers.
A limitation of using multi-class classifiers is the requirement of representative training examples from legitimate and bot accounts. Although the examples from the legitimate class are relatively easy to obtain, the collection of representative examples from bot accounts is not a trivial task, as these need extensive verifications to assure the account is not of a legitimate user [3]. Furthermore, Cresci et al. [4] have recently proposed that there exist different types of bots and that each type has its characteristic behavior. These new types of bots could be challenging to detect by supervised classifiers if the behaviors found in the training examples are too different. Taking into account that the type of bots will continue to evolve in the future, with bot creators modifying the behaviors to avoid detection, a new strategy for automatic bot detection is needed.

Online social networks (OSNs) are popular platforms that connect users all over the globe. A botnet represents a group of agents (bots) that are managed and programmed to act in an organized manner. The term social botnet refers to a new generation of botnets that utilize OSNs as command and control (C&C) channels with minimal noise. A social botnet can be used for disinformation campaigns without being detected as pieces of software. Social bots can also be utilized for benign purposes. For instance, approximately 15% of all edits on Wikipedia are made by bots [5]. However, even benign types of social bots can sometimes be harmful; for example, through the spreading of unverified information, which can be considered a result of the lack of fact-checking in most automatic trading systems. Nappa et al. designed a botnet using Skype protocol, which includes fault-tolerant, encrypted communication. Moreover, it is

firewall- and network address translation (NAT)-agnostic, which means that it does not require any additional network configuration. Koobface is another example of a malware program that has proved successful in propagating through different social networks. The OSN propagation components of Koobface included several binaries, each of which has been developed to handle a particular OSN.
In August 2014, Twitter reported that approximately 8.5% of its users are bots [6]. Moreover, Zhang & Paxson (2011) showed that 16% of these bots exhibit highly automated behavior. Recently, Cresci et al. [7] conducted a large-scale analysis of spam and bot activity in stock microblogs on Twitter. Their findings show that 71% of retweeting users were classified as bots, 37% of which had been suspended by Twitter. The Defense Advanced Research Projects Agency (DARPA) conducted a four- week competition in February and March 2015, where the challenge was to identify influence bots on Twitter [6] Competitors had to:
(1) Separate influence bots from other types of bots;
(2) Separate influence bots that are related to a specific topic t from those related to other topics; and
(3) Separate influence bots related to topic t that seek to spread sentiment s from those that are either neutral or that spread the opposite sentiment polarity.
Six teams (University of Southern California, Indiana University, Georgia Tech, Sentimetrix, IBM, and Boston Fusion) were involved in the contest to detect 39 influence bots; however, the teams were given no prior information about the number of bots included in the competition. The results of DARPAs botnet challenge suggest that a bot-detection system needs to be semi-supervised; that is, human judgments are required to improve detection of these bots. Visualization methods are also necessary to help OSN experts make better decisions regarding suspicious bots.

1.1 Social Bot Profiling

Many studies in the literature in this field focus merely on distinguishing between legitimate users and malicious social bots. Other studies, however, suggest more fine-grained classification at different levels. For instance, Kartaltepe et al. [8], in their client-side countermeasure, collected process-level information to distinguish between benign and suspicious processes. In social graph-based defense approaches, bots are often viewed as Sybils (fake accounts) that are mainly used to forge other users' identities and launch malicious activities. Cao et. al. [9] refers to the Sybils adjacent to the attack edges as entrance Sybils, and to other Sybils as latent Sybils. Entrance Sybils represent well-maintained fake accounts. These have lower rejection rates than latent Sybils, which send requests to random non-Sybil users.
Social bots can also be further categorized into different fine-grained classes. Chu et al. [9] defined a bot as an automated account used for spam purposes, and a cyborg as a bot-assisted human or human-assisted bot. Tavares & Faisal [10] categorized Twitter accounts into three groups:
(1) Personal,
(2) Managed, and
(3) bot-controlled accounts based on their time intervals between tweets.

Even though they may be involved in malicious activities, no real users' accounts (such as human spammers, trolls, managed accounts) are defined as social bots. Stringhini et al. [11] also defined different types of spambots displayers, braggers, posters, and whisperers, while Clark et al. [12] focused on three distinct classes of automated tweeting: by robots, cyborgs, and human spammers. El-Mawass et al. [13] studied verified users (cyborgs), human users, trend hijackers, and promotional spambots. Abokhodair et al., [14] classified each bot into one of five categories (short-lived bots, long-lived bots, generator bots, core bots, and peripheral bots) based on distinct behavioral patterns. Song et al. [15] analyzed retweets generated by three account groups from black market sites:
(1) Normal,
(2) Crowdturfing, and
(3) Bots.
Fernquist et al. [16] used the term bot to refer to accounts that show automated behavior, including automated social bots and sock puppets. Gao et al. distinguished between spam and legitimate clusters. Studies have also highlighted that honey profiles can be used to capture the behavioral patterns of social bots. For instance, by taking a honey-profile-based approach, Stringhini et al. were able to identify two kinds of bot behaviors greed and stealth. Greedy bots add spam content to every message they send, whereas stealthy bots send messages that look legitimate, and inject spam content only once in a while. As an alternative approach, Freitas et al. used a reverse-engineering technique to understand the different infiltration strategies of malicious social bots on Twitter. Alongside the categories mentioned above, which mainly deal with malicious bots, Oentaryo et al. (2016) provided a new categorization that includes both benign and malicious bots. This behavior-based system consists of three main categories broadcast, consumption, and spambots. A broadcast bot is managed by a particular organization or a group and is mainly used for information dissemination purposes. Consumption bots are used to aggregate content from different sources and provide update services, while spambots are used for delivering malicious content.
As we have discussed above, social bot profiling is an essential step for many reasons:
(1) Detection approaches may focus on certain types of benign or malicious social bots while missing other types,

(2) OSN administrators need to allow benign bots while mitigating against malicious ones, and
(3) Other benign accounts such as cyborgs and managed accounts can be falsely detected as malicious bots, yielding undesirable high false-positive rates.

II. RELATED WORK

2.1 Bot on a Twitter account:

Since our research is to do with detecting the presence of a bot on a Twitter account, we shall focus this section's discussion on related research for bot detection in that social network. Bots on Twitter are used with multiple motivations in mind. One group of bots, known as *spammers* or *content polluters*, are used to distribute spam, attract customers to products, generate revenues, and disseminate malicious websites/programs. Other groups are used to increase the popularity of an account, by adding followers or generating conversations about it, and to increase the influence the account has on the social network. Another group is used for political purposes, to influence the population perception about a candidate, fake a grassroots movement, and to infiltrate social discussions to manipulate them. Given the multiple uses of bots, it is a paramount problem to find methods to detect them.

Bot detection on Twitter is based on the premise that a genuine, human account displays a behavior that is different from the one output by a machine, regardless the machine level of automation or sophistication. Commonly, to characterize human intervention behind a Twitter account, detection models make use of features extracted from five elements: the content of tweets posted from the account, how the poster is thought to feel about the topic contained in a tweet, general information about the account, account activity, and the associated account social network. We will respectively refer to these kinds of features as: tweet content, tweet sentiment, account information, account usage, and social network structure. We outline these kinds of features below.

Tweet Content: a tweet content feature is one related with the message of a tweet and is obtained through message parsing analysis. Among others, it could be the number of words in a tweet, the use of capitalization and punctuation, the number and type of a sequence of *n* letters (called an *n*-gram), and whether the text resembles a known spam message. Moreover, tweet content features also include special elements, such as the number of URLs, hashtags, or mentions to other users.

Tweet Sentiment: a sentiment feature is a piece of information extracted by applying sentiment analysis to the text of a tweet. This is used to determine how the poster feels about the topic of the associated tweet. We have separated this feature type from tweet content, as the feature is about how the poster feels, and not the text form per se. Sentiment features aim to measure different sets of sentiments about a topic, and the degree expressed to each sentiment. Sentiment analysis techniques are also used to compute whether the poster agrees or not with a topic, and the strength of such agreement.

Account Information: an account information feature is extracted from the Twitter account that has been allegedly used to post a message. Features of this kind involve account creation date, Twitter username, account description, account language, account profile picture, description URL (if the account is verified), number of friends, number of followers, and the ratio between the two latter figures; among others. Since these features do not tend to change frequently, they do not need to be extracted with the same regularity than tweet content, or tweet sentiment features.

Account Usage: Account activity includes any measurement as to how regularly an account is used to post a tweet, the similarity between several tweets posted from the account, and how the user makes use of Twitter (for example, via a mobile device, a web interface, or an automated tool). In this type of feature, there are also statistical metrics computed from some of or all of the tweets in the account; these involve, for example, the number of different URLs and hashtags posted. Commonly, these statistical features are represented as a ratio between the number of occurrences of a distinctive element and the number of tweets.

Social Network Structure these features are used to measure message flow and account interaction. Here, we may have, for example, inter-account post similarity (in terms of content), or other typical behavior, including the number of replies or retweets, on a small-time period, to a single tweet, the number of bidirectional links between accounts (where a client's friend is also a client's follower), and general sentiment agreement about a topic in a group of accounts. The use of these features entails the observation of the behavior of multiple related accounts.

Twitter bot detection has been approached using a combination of these features, and, as expected, these appeared over time, being tweet content the earliest feature type supplied to a classifier. Bot detection has been approached using both types of learning: supervised and unsupervised. In what follows, we shall discuss the most prominent bot detection mechanisms; in passing, for each mechanism, we also survey the feature space it has made use of, and the performance it has claimed to have attained.

However, the reader should bear in mind that performance figures cannot be fairly compared because authors have used different experimental setups.

2.2. Supervised Approach to Bot Detection

The earliest supervised methods for bot detection work under the premise that bots have more friends than followers, post more URLs, mention more users (whether friends or not), and post very similar messages. Lee et al. [17] proposed a classifier, called Decorate, which makes use of features of the following types: tweet content, tweet account, and account usage. Authors showed that *Decorate* achieved a performance of 0.88 using the F1 measure; in their experiments, [17] considered a dataset developed by their own. Later, Wang [18] proposed a method that adopts a similar approach, but it focuses only on the last 20 tweets issued by the account under observation. Using a naïve Bayes classifier, Wang [18] has reported on to have achieved an F1 equal to 0.91. Later, Ahmed and Abulaish tested the performance of naive Bayes, J48, and Jrip (taken from Weka data mining tool), when each classifier is restricted to different feature subsets. They found that account usage features are more discriminative than tweet content ones.

Dickerson *et al.* [19] reported that using sentiment features, in a social network-based Random Forest, improves Bot detection. After, Wang et al. proposed to use tweet content, sentiment, tweet account, and tweet usage features. Wang showed that using a Random Forrest on the dataset created by [18], they obtained an F1 equal to 0.94. However, [18] reported to have obtained a result contradicting to [19]: using features that are either tweet or tweet usage only they obtained an F1 of 0.9.

Davis *et al.* [14] proposed a bot detector (called Bot or Not) that combines all the features previously mentioned. Bot or Not uses a Random Forest classifier over 1,000 features from all the categories, obtaining a 0.95 of Area under the ROC Curve (AUC). It is important to highlight that Bot or not has an API available to the public. Later, Cresci *et al.* [20] proposed that not only are the actions but the sequence of the actions which allows discerning if an account is a bot. They use a DNA-like analysis, where they assign a letter to each tweet on an account, depending on the presence of specific tweet content features: if the tweet has an URL, mentions, hashtags, images, a combination of these, or none of these. Then, the DNA fingerprint of an account is the string with all the assigned letters of each tweet, appearing in order of posting. They hypothesize that bots should have longer common substrings than normal users do. To test this, they propose to calculate the length of the longest common substrings and consider those accounts with a value higher than a threshold as bots. They report an F1 of 0.97 using this approach. Nevertheless, it is well-known that the DNA-like analysis approach has high computational complexity.

2.3 Social Bot Detection Methods

In this section, we focus on social network-based detection of malicious bots, with the detection approaches falling into three main categories. We also investigate the strengths and weaknesses of these approaches, along with their effectiveness and shortcomings in real-world OSNs.

2.3.1. Graph-based approaches

As mentioned previously, in social graph-based defense approaches, the term bots often refer to Sybils (fake accounts), which are mainly used to forge other users' identities and launch malicious activities [40]. Graph-based approaches are based on three key assumptions. First, the cut between the Sybil and honest regions is sparse. Second, the benign region is of a fast-mixing nature, which implies a high probability that a random walk of steps will end in a random edge in the honest region. Third, the social edges represent strong trust relationships, and therefore it is hard for attackers to create links with legitimate users (i.e., with the honest region).

2.3.2. Strong assumption-based approaches

These approaches rely on satisfying all of the above-mentioned assumptions, which may not reflect real-world OSNs. Strong assumption-based detection approaches fall into the following two categories.

A) Random-walk-based approaches

Sybil Guard [21] performs long random walks that should intersect with the suspect node in order to accept it as honest. It makes use of the following observations, which are applicable only when the number of attack edges is limited:

(1) The average honest nodes random route is highly likely to remain within the honest region, and

(2) Two random routes from honest nodes are highly likely to intersect within the random walk.

SybilLimit [21], on the other hand, uses multiple instances (pm) of short random walks to sample nodes from the honest set and accepts a node as honest when both:

(1) There is an intersection among the last edge of the random routes of the verifier and the suspect, and

(2) The intersection satisfies a balance constraint.

Accordingly, one can conclude that SybilGuard performs intersections on nodes, whereas SybilLimit applies intersections on edges. Both SybilGuard and SybilLimit are highly vulnerable when high-degree nodes are compromised. Therefore, it is implied that these two approaches are more effective in defending against malicious users than defending against compromised honest users. They also include a pre-processing step, in which nodes with degrees smaller than a predefined threshold are removed. As a result, a large number of nodes will be pruned due to the fact that social networks often have a long-tail degree distribution. Moreover, according to how the OSN operator treats these pruned nodes, this process can result in high false-positive/-negative rates [22]. Fast mixing with the honest region implies the absence of small cuts, whereas the slow mixing between honest and dishonest implies that one can compute the bottleneck cut between honest and Sybil nodes to infer the labels of each node. SybilInfer considers the

disturbance in fast mixing between the honest and dishonest regions as a faint bias in the last node of a short random walk. Unlike SybilGuard and SybilLimit, SybilInfer depends primarily on the number of colluding malicious nodes, rather than on the number of attack edges, and it also performs independently of the topology of the Sybil region. SybilResist [23] consists of four main stages:
(1) Performing random walks _ (log n) starting from a known node, followed by selecting the nodes with a frequency higher than a predefined threshold, which are then treated as honest nodes;
(2) Sybil identification to identify the suspect nodes based on comparison with the mean and standard deviation of the previous random walks;
(3) walk-length estimation to set the initial value for Sybil region detection algorithms, and
(4) Detecting the Sybil region among the detected nodes.
The main drawback here is that, every time the structure of the Sybil region changes, the performance changes according to the lengths of the random walks used in each step. Furthermore, SybilResist is also highly vulnerable when high-degree nodes are compromised.

2.3.3 Relaxed assumption-based approaches

First, the previous approaches bound the accepted Sybils to the number of attack edges based on social graph properties and structure, as the third assumption suggests that the number of attack edges is limited. [23] Showed that RenRen, the largest social networking platform in China, does not follow this assumption, which also implies that real-world social networks may not necessarily represent strong trust networks.
Second, [24] examined the link-farming phenomenon on Twitter and showed that specific users, called social capitalists, have a tendency to follow back any account which follows them, in order to promote their content and to increase their social capital. Spammers, on the other hand, can exploit this behavior to farm links and promote their malicious content. This phenomenon can be considered as further evidence that real-world social networks may not necessarily represent strong trust networks; therefore, the honest region may not be easily separable from the Sybil region.

2.3.4 Weighted trust propagation-based approaches

The approaches discussed in this section propagate trust from a weighted social graph via power iterations (i.e., a PageRank-like approach). SybilFence makes use of the observation that most fake users receive a large amount of negative feedback from legitimate users, such as rejections of their friend requests. SybilFence reduces the weights of social edges for users who have received negative feedback, which, in turn, limits the impact of Sybil's social edges. SybilFence adapts the SybilRank approach in its weighted defense graph. Its method consists of three main steps:

- Trust propagation,
- Trust normalization, and
- Ranking users based on their degree-normalized trust.

SybilFence is resilient to Sybils flooding requests and also outperforms SybilRank using the negative feedback of legitimate users. However, SybilFence assumes that the non-Sybil region is well connected after social edges are discounted, which can be an unrealistic assumption in real-world OSNs such as RenRen.

2.3.5 Loopy belief propagation-based approaches

In Sybil Belief, [25] utilizes information from a small set of known benign and/or Sybil users. It does not use random walks; rather, it adopts a semi-supervised learning approach, in which the goal is to propagate reputations from a small set of known benign users and/or Sybils to other users along the social connections among them. Sybil Belief mainly relies on Markov random fields (MRFs) and loopy belief propagation (LBP), and its performance is better by orders of magnitude than that of SybilLimit and SybilInfer in terms of both the number of accepted Sybil nodes and the number of rejected benign nodes. However, the number of accepted Sybil nodes increases dramatically when the labeled benign and Sybil nodes are highly imbalanced. Sybil Belief adopts LBP to make inferences about the posterior information. However, for social networks with loops, LBP approximates the posterior probability distribution without theoretical convergence guarantees. Another limitation is that LBP-based methods are sensitive to the number of iterations that the methods run.

III. PROPOSED ALGORITHM

Proposed the Random Forest algorithm is an ensemble learning algorithm, meaning the algorithm is actually made up of many other basic machine learning algorithms. To predict a class, each basic machine algorithm votes for a class and after all of the basic algorithms have voted, the class with the most votes is the class the ensemble algorithm predicts. With Random Forests, the underlying algorithm used is the Decision Tree classifier, hence the "Forest" in Random Forest. The Random Forest algorithm brings randomness into the model when it is growing the Decision Trees. Instead of searching for the best attribute while splitting a node, or decision in the tree, it searches for the best attribute among a random subset of attributes. This process creates diversity among trees and allows for each tree to be built upon different attributes, which generally results in a better model. We choose the Random Forest algorithm because of its general effectiveness when classifying categorical inputs.

3.2 Detailed Working:
Detailed working of the proposed system is shown below.

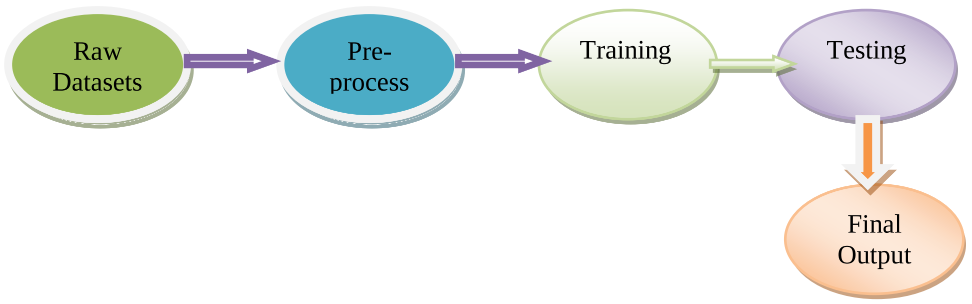


Figure 3.2: Proposed Architecture.

3.3 Proposed Classifier Algorithm
Proposed Classifier Algorithm is as follows:

Step 1: Read in bot data from our data set.

Step 2: Read in user data from our dataset. Make sure that all the keys for each user is ordered the same way then convert the dictionary to a list of values. This is for classification. Having data in the wrong index can throw-off the classifiers during training.

Step 3: Convert the dictionaries of bots and users to lists.

Step 4: Shuffle the data.

Step 5: Convert the bot and user lists to numpy arrays.

Step 6: Split the data into 10 folds (for cross validation).

Step 7: train and classify with a random forest.

7.1: segment the data into training and testing sets

7.2: merge the training bot and user data into a training set

7.3: generate training labels

7.4: merge the testing bot and user data into a testing set

7.5: generate testing labels

7.6: Convert the test Labels to a numpy array and compute the accuracy for this iteration. Then add the accuracy to the list of accuracies for each classifier

Step 11: Average the accuracy rates (for each classifier) and convert the accuracy rates to percent

Step 13: End Algorithm.

Our classification approach was to use the variety of classification methods provided by scikit-learn library in python to determine the best classifier that would most accurately determine if a Twitter profile was a bot or user. We used Random Forest, K-nearest neighbor, Naive Bayes, Multi-Layer Perceptron, Logistic Regression and SVM to classify our dataset and incorporated bagging which improved accuracy of all classifiers with the exception of naive Bayes and random forest. We took 50% of the features and 50% of the training data to train each bagging classifier. This pushed the all the classifier's accuracy rates above 85% but without bagging gave accuracy rates in the 60's and 70's for KNN, SVM, and Naive Bayes. For most of the classification methods, we used the default parameters provided by the library with a bit of fine tuning. Finally, we incorporated K-fold cross validation with K equal to 10 to obtain the average accuracy of the classifiers.

● Random forest classifier used 100 decision trees that used the gini index to determine the measure of impurity of a split.

● Decision tree classifier also used gini index to determine the measure of impurity of a split to determine the best attribute to split at. For example, in our decision tree model, we found that the best split was the "replies" attribute (Visualizations).

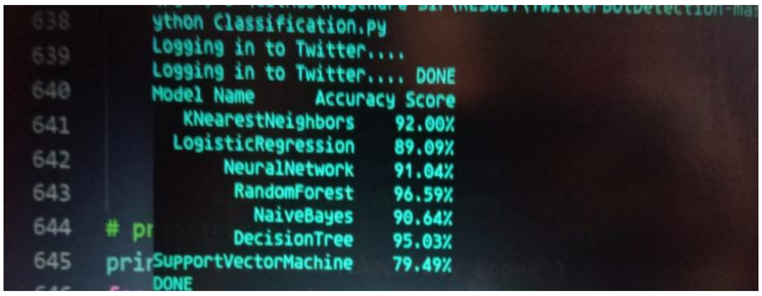
● K Nearest Neighbor classifier with K equal to 4 in conjunction with Euclidean metric to calculate distance and bagging to improve accuracy.

● The default Bernoulli Naive Bayes was used with Laplace smoothing to avoid the Zero probability problem and without bagging. The Naive Bayes method binarized the values of each feature provided since it only handles binary-valued feature vectors. Since we determined the features we are using by intuition, Naive Bayes is a method that would handle the irrelevant attributes that we had chosen if there was one. We also wanted a classifier which used independence assumption to assume all our features are independent from each other which mean each attribute is estimated in one dimensional space to see how well the classifier does compared to other classifiers. This is a pro and a con since it avoids the curse of dimensionality that other classifiers might run into but it doesn't consider the correlation between the attributes such as the correlation between tweet frequency and lexical diversity.
● MLP classifier using L-BFGS as the solver for weight optimization with bagging to improve accuracy. We set the seed for the random number generator to be 1 and (1e - 5) for the "L2 penalty (regularization term) parameter". Lastly the parameters for hidden_layer_sizes we set as (7, 4) to fit our 7 features and to have a 2-hidden layer neural network.
● Default linear regression with 1e5 for the inverse of the regularization strength with bagging to improve accuracy.
● Linear support vector classifier that used square of the hinge loss as the loss function and bagging to improve accuracy. We used a linear version because it works better with a larger sample size as compared to the regular support vector machine.

IV. SIMULATION RESULTS

Some important findings regarding the classifiers we used also surfaced throughout our experiments. We noticed that naive bayes perform worse with bagging than they do without bagging. Bagging increases the predictive error for naive bayes and it doesn't do much to reduce variance for naive bayes because it is already a low variance classifier. Furthermore, since bagging only takes a sample of the training set to train each naive bayes classifier this contributes to predictive error since the classifier doesn't have all the prior knowledge it needs of actual probabilities per attribute to determine if a profile is a bot or human. Support vector machine performed the worst because it had difficulty finding the best hyperplane to separate the data points given there were seven attributes and data points might have become sparse when finding such a cut in higher dimensions.

5.2 Results & Evaluation



```
ython Classification.py
Logging in to Twitter....
Logging in to Twitter.... DONE
Model Name         Accuracy Score
     KNearestNeighbors    92.00%
    LogisticRegression    89.09%
         NeuralNetwork    91.04%
          RandomForest    96.59%
            NaiveBayes    90.64%
          DecisionTree    95.03%
 SupportVectorMachine    79.49%
DONE
```

Figure 5.1: Output of proposed and existing classifiers.

Evaluations of various classifier algorithms according to accuracy are displayed below:

Table 5.1: Performance Evaluation

Method	Testing Accuracy (%)
Support Vector Machine	79.49
Logistic Regression	89.09
Naive Bayes	90.64
Neural Network	91.04
KNN	92.00
Decision Tree	**95.03**
Proposed Method	**96.59**

V. CONCLUSION AND FUTURE WORK

The two main contributions from this thesis are:

- The first contribution is the bot/human classification algorithm we developed.
- The second contribution from this thesis is the analysis and rationale for the features we use. Extending this point, we also contribute some possible features that we suspect will increase accuracy and better handle the bot population as a whole.

- All things considered, our algorithm's performance meets the expectations we had at the set out of this project.

5.1 Future work

There is still much work that could be done to further improve this algorithm. Since this thesis can effectively be split into three sub-steps, it follows naturally to analyse the improvements that can be achieved in each of these sub-steps. The first step is sourcing training/testing data. Since we would like to see how our algorithm behaves in a multitude of environments, we intend to collect more and different data. In the introduction we presented a scale depicted how bots have various complexities. As we already stated, for this study we decided to focus on middle tier bots. However, since we did not train our classifiers on data containing top tier and low tier bots, we cannot accurately classify those accounts. Collecting low tier bots is fairly simple, since these bots may easily be purchased on the internet for a small fee. The real difficulty in improving our data will lie in collecting top tier bots. These bots are purposely disguised to behave like humans, since their objective is generally to influence some cause. Since bots are generally going to be focused on some theme, we plan to perform our own Turing tests on tweets coming from accounts tweeting about trending topics. However, this only pertains to finding accounts and collecting their screen names. The other aspect of data collection is actually sourcing the data for each of these users. As mentioned in the previous sections, we collect data from Twitter via the Twitter API. However, when there is a large volume of data to source from Twitter, data collection can take a lot of time. To make our algorithm more efficient at data collection, we plan on taking advantage of the multiple cores on our systems by multi-threading the process.

REFERENCES

[1] Cresci S, Petrocchi M, Spognardi A, Tognazzi S. Better Safe than Sorry: An Adversarial Approach to Improve Social Bot Detection. 2019; Available from: http://arxiv.org/abs/1904.05132.
[2] Pantic N, Husain MI. Covert botnet command and control using Twitter. In: Proceedings of the 31st Annual Computer Security Applications Conference. ACM; 2015. p. 171–180.
[3] Yang C, Harkreader R, GU G. Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers. IEEE Transactions on Information Forensics and Security, 2013 Aug; 8(8):1280–1293.
[4] 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. International World Wide Web Conferences Steering Committee; 2017. p. 963–972.
[5] Steiner, T. (2014). Bots vs. wikipedians, a nons vs. logged-ins (redux): A global study of edit activity on Wikipedia and wikidata. In Proceedings of the International Symposium on Open Collaboration (p. 25). ACM.
[6] Subrahmanian, V. S., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., Zhu, L., Ferrara, E., Flammini, A., & Menczer, F. (2016). The darpa twitter bot challenge. Computer, 49, 38–46.
[7] Cresci, S., Lillo, F., Regoli, D., Tardelli, S., & Tesconi, M. (2018b). Cashtag piggybacking: uncovering spam and bot activity in stock microblogs on twitter. ArXiv preprint arXiv: 1804.04406.
[8] Kartaltepe, E. J., Morales, J. A., Xu, S., & Sandhu, R. (2010). Social network-based botnet command-and-control: emerging threats and countermeasures. In International Conference on Applied Cryptography and Network Security (pp. 511–528). Springer.
[9] Cao, Q., & Yang, X. (2013). SybilFence: Improving social-graph-based Sybil defenses with user negative feedback. ArXiv preprint arXiv: 1304.3819.
[10] Chu, Z., Gianvecchio, S., Wang, H., & Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? IEEE Transactions on Dependable and Secure Computing, 9, 811–824.
[11] Tavares, G., & Faisal, A. (2013). Scaling-laws of human broadcast communication enable distinction between human, corporate and robot twitter users. PloS one, 8, e65774.
[12] Stringhini, G., Kruegel, C., & Vigna, G. (2010). Detecting spammers on social networks. In Proceedings of the 26th Annual Computer Security Applications Conference ACSAC '10 (pp. 1–9). ACM.
[13] Clark, E. M., Williams, J. R., Jones, C. A., Galbraith, R. A., Danforth, C. M., & Dodds, P. S. (2016). Sifting robotic from organic text: a natural language approach for detecting automation on twitter. Journal of computational science, 16, 1–7.
[14] El-Mawass, N., Honeine, P., & Vermonter, L. (2018). Supervised classification of social spammers using a similarity based markov random field approach. In Proceedings of the 5th Multidisciplinary International Social Networks Conference (p. 14). ACM.
[15] Abokhodair, N., Yoo, D., & McDonald, D. W. (2015). Dissecting a social botnet: Growth, content and influence in twitter. In Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing CSCW '15 (pp. 839–851). New York, NY, USA: ACM.
[16] Song, J., Lee, S., & Kim, J. (2015). Crowd target: Target-based detection of crowdturfing in online social networks. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 793–804). ACM.

[17] Fernquist, J., Kaati, L., & Schroeder, R. “Political bots and the Swedish general election”, In 2018 IEEE International Conference on Intelligence and Security Informatics (ISI) (pp. 124–129). IEEE-2018.
[18] Haustein, Stefanie, et al. "Tweets as impact indicators: Examining the implications of automated “bot” accounts on Twitter." Journal of the Association for Information Science and Technology 67.1 (2016): 232-238.
[19] O. Loyola-Gonzalez, A. Lopez-Cuevas, M. A. Medina-Perez, B. Camina, J. E. Ramirez Marquez, and R. Monroy, “Fusing pattern discovery and visual analytics approaches in tweet propagation”, Inf. Fusion, vol. 46, pp. 91_101, Mar. 2019.
[20] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi,” The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in Proc. 26th Int. Conf. World Wide Web Companion (WWW), Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 963-972.
[21] Yu, H., Kaminsky, M., Gibbons, P. B., & Flaxman, A. (2006). SybilGuard: defending against Sybil attacks via social networks. In ACM SIGCOMM Computer Communication Review 4 (pp. 267–278). ACM.
[22] GAO, P., Wang, B., Gong, N. Z., Kulkarni, S. R., Thomas, K., & Mittal, P. (2018). Sybilfuse: Combining local attributes with global structure to perform robust Sybil detection. In 2018 IEEE Conference on Communications and Network Security (CNS) (pp. 1–9).
[23] Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B. Y., & Dai, Y. (2014). Uncovering social network Sybils in the wild. ACM Transactions on Knowledge Discovery from Data (TKDD), 8, 1–29.
[24] Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Korlam, G., Benevenuto, F., Ganguly, N., & Gummadi, K. P. (2012), Understanding and combating link farming in the twitter social network. In Proceedings of the 21st International Conference on World Wide Web WWW ’12 (pp. 61–70). ACM.
[25] Gong, N. Z., Frank, M., & Mittal, P. (2014). Sybil belief: A semi-supervised learning approach for structure-based sybil detection. IEEE Transactions on Information Forensics and Security, 9, 976–987.