IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

A New Approach to Bot Detection: Striking the Balance Between Precision and Recall

Fred Morstatter, Liang Wu, Tahora H. Nazer
Arizona State University
Tempe, Arizona, 85281
{fred.morstatter, wuliang, tahora.nazer}@asu.edu

Kathleen M. Carley
Carnegie Mellon University
Pittsburgh, Pennsylvania, 15213
kathleen.carley@cs.cmu.edu

Huan Liu
Arizona State University
Tempe, Arizona, 85281
huan.liu@asu.edu

***Abstract*—The presence of bots has been felt in many aspects of social media. Twitter, one example of social media, has especially felt the impact, with bots accounting for a large portion of its users. These bots have been used for malicious tasks such as spreading false information about political candidates and inflating the perceived popularity of celebrities. Furthermore, these bots can change the results of common analyses performed on social media. It is important that researchers and practitioners have tools in their arsenal to remove them. Approaches exist to remove bots, however they focus on precision to evaluate their model at the cost of recall. This means that while these approaches are almost always correct in the bots they delete, they ultimately delete very few, thus many bots remain. We propose a model which increases the recall in detecting bots, allowing a researcher to delete more bots. We evaluate our model on two real-world social media datasets and show that our detection algorithm removes more bots from a dataset than current approaches.**

I. Introduction

Social media is an important part of every day life. With billions of users producing and consuming information every day, it is a natural extension that people turn to this medium to read and disseminate news. With so many people turning to social media, malicious users like bots have begun using these sites to sway the discussion [15], [16], [18]. Bots, social media accounts that are controlled by software, have risen to prominence on social media. Bots cause users to lose trust that social media platforms can deliver news honestly, as they become suspicious that the stories they see at the top of their feeds were "pushed" there by manipulative bots. Furthermore, bots impinge the work researchers perform on social media as they can cause researchers to draw false conclusions about the populations under study. Researchers wish to understand human behavior through the lens of social media [14], and this is often impinged by the wealth of content pollution created by automated social media accounts.

Bot detection is an important task in social media. Twitter, a popular social media platform, is plagued by automated accounts. One study has estimated that over half of the accounts on Twitter are not human[1]. More conservative studies estimate that the number of bots on Twitter lie somewhere between 5-9% of the overall population[2]. These bots, generate 24% of the tweets produced on Twitter[3]. While some bots innocuously tweet the time of day or a record of historical events, the ones we detect in this study seek to actively sway the discussion on social media by promoting certain trends and retweeting particular users. These bots are considered malicious as they try to perturb the discussion on social media, swaying the discussion to the topics of their choosing by falsely amplifying the size of a topic or keyword.

Bots are not just a danger to the users of social media, but also to those that study it. By coordinating massive tweeting campaigns of particular hashtags, bots have been able to influence measures on Twitter, including the trending topics [16]. This manipulation of Twitter also bleeds across into the analysis that is done on this particular platform. Bots can also influence statistics performed on Twitter data, such as the top hashtags and the most important users in the data. This means that bots not only have the potential to damage the active conversation on social media, but they can also cause additional destruction by harming the quality of the work done by researchers studying social media. Clearly, there is a need for us to remove bots from social media both for the benefit of users and researchers.

Current approaches to bot detection focus on precision [13], [16], [23]. Optimizing precision can be useful as it helps to avoid the possibility of a real user being deleted from the site, which may anger other real users or cause them to leave the site. However, this is one extreme. An algorithm that only considers precision will have many false-negatives, as it will only predict the most acute examples as bots, leaving many bots undetected. This is not ideal from a research perspective, where we wish to study human behavior on social media. Thus, we want to include recall as the objective of our bot detection model. However, this is another extreme. The major concern with optimizing recall is that it is trivial to achieve a perfect score: simply marking every user as a bot will yield a recall of 100%. Thus, we propose to study the balance between these measures. While our model focuses on recall, we measure its performance using the F_1 score.

In this work, we propose a model that can detect bots on social media with a focus on recall. We evaluate by taking into account both precision and recall to strike the balance between these important evaluation measures. In this way,

[1] http://blogs.wsj.com/digits/2014/03/21/new-report-spotlights-twitters-retention-problem/

[2] http://www.nbcnews.com/technology/1-10-twitter-accounts-fake-say-researchers-2D11655362

[3] https://sysomos.com/inside-twitter/most-active-twitter-user-data

IEEE/ACM ASONAM 2016, August 18-21, 2016, San Francisco, CA, USA
978-1-5090-2846-7/16/$31.00

we show that our method can help to remove the most bots, while simultaneously keeping the precision relatively high. By applying our work, researchers can remove more bots from their social media dataset and focus on the messages produced by real users.

A. Bot Definition

A bot is any automated account in an online social network. This definition differs slightly from the one found in [1] and [2] as they differentiate social bots from self-declared and spambots although socialbots and spambots may have the same structure and purpose.

B. Problem Statement

Given a set of all of the users in a dataset, U, and a set of labeled bots $b \subset U$, identify a set of users $u \subset U$ that maximizes the function:

$$F_1(u, b) = 2\frac{PR}{P + R} \tag{1}$$

where $P = \frac{|u \cap b|}{|u|}$ is the precision of the model, and $R = \frac{|u \cap b|}{|b|}$ is the recall.

The main contributions of this paper are as follows:

- We collect two bot datasets using state-of-the-art collection techniques, and test two labeling techniques. We publish this data to allow for reproducibility and encourage comparison with future work.
- We build a model that detects bots on Twitter with an emphasis on the F_1 measure, which considers recall as well as precision. We compare this model to existing approaches for bot detection and find that the model achieves superior performance in this task, yielding high recall with only a minor loss in precision, which contributes to it achieving the best F_1 score in all algorithms we studied.

The rest of this work is organized as follows. First, we discuss the related work to this problem. Next, we introduce the two datasets we collected for this work. We continue to propose a novel bot detection method, $BoostOR$, which optimizes the F_1 score to remove the most bots from a given dataset while considering precision. Then, we test our model and show that it is able to remove more bots than other methods. Finally, we provide discussion and conclude.

II. Related Work

Bot detection approaches in general try to build a classifier that labels a given user as human or bot. In the first part of this section we introduce different detection methods and group them based on features they extract from the data to train the classifier. Then we discuss techniques of ground truth acquisition, their distinctions, pros and cons. Finally, we enumerate different evaluation mechanisms that have been used for this task.

Recent statistics show that more than 50% of Twitter accounts are not human users. Social network administrators are well aware of these harmful activities and try to delete these users using their suspension/removal systems. By one estimate 28% of accounts created in 2008 and half of the accounts created in 2014 have been suspended by Twitter [9]. What is not well taken care of is the role of bots in facilitating these malicious activities. In one study, 145,000 accounts survived for months without detection. Today, 16% of spammers in Twitter are bots [6].

Although social network spam detection approaches are still insufficient, bot detection in social networks has received wide attention from the research community in recent years [4], [7], [8], [10], [17]. Methods proposed to solve this problem mostly observe it as a classification task which extracts features from the user in order to classify it as a bot or a human. A categorization of the features used in these classifiers is as follows:

- *Content of posts and messages:* In using content, the goal is to find properties that show the difference between what is published by a bot and what is published by a normal user. The text sent by each bot is significantly different from another bot but they all tend to use URLs to link to the content they promote [23]. Although the content is specific to each bot, they all have high tendency in using URLs for promotional purposes. Furthermore, extracting the sentiment of tweets [16], [15], language and length of messages [20], number of URLs [3], [12], [21], number of mentions and similarity between all pairs of tweets posted by a user [12], and originality of tweets [21] are all examples of using content.
- *Profiles and activities:* Automation involved in generating bot accounts results in profile properties that contain detectable patterns. Bots that were working together towards political disruption in 2011 Russian parliamentary election were found to have similar email addresses and account creation times [18].
  Bots mostly tweet using automatic devices [3], have a shorter life time [12], short subscriber details [4], and use hijacked IP addresses [18]. In an effort to escape from suspension systems, the controller of bots spreads the machines from which he launches his attacks geographically [8].
  Even when profile information is not available, the username alone can be a bot indicator. The length of the screen name [12] and the distance of the unigram/bigram distribution from verified names [13] can differentiate bots from normal users. Matching how well the screen name matches human typing patterns can also differentiate automated users [25].
- *Network structure and connections:* To boost their desired effect, Twitter bots follow normal users hoping to be followed back. To this aim they show mass following and unfollowing behaviors and thus their connections show different characteristics. The number of followers, number of friends, ratio of the the friends to followers, percentage of bidirectional friends, and the standard deviation of unique numerical IDs of followers and friends have been all used to represent this [12].

In order to build a classifier, we need to obtain a gold

standard dataset: a set of users and their bot/human label to evaluate the results. Of course, this information does not come affixed to each user when the researcher collects the dataset. Thus, the onus is on the researcher to collect these labels. There are three main approaches to this:

- *Manual annotation:* As in most other classification problems, manual labeling can be to obtain ground truth. Although this method has been used widely in this field [3], [4], [6], [15], [23], we still have the challenge of how to choose annotators and how many annotators are sufficient in order to achieve reliable labels. Furthermore, this method does not scale as it takes time to recruit users and funds to pay them.
- *Suspended users lists:* This method leverages the social networking site itself to obtain the labels. The researcher simply observes the users and sees which ones get suspended or removed by the site. This approach is simple, but the concern remains that the reason of suspension and deletion is not usually indicated in such lists. Thus, many suspension lists include users who violated other rules and regulations of the site. Methods proposed in [7], [13], [18] use these lists.
- *Honeypots:* A newer ground truth acquisition method is based on using honeypots, bots created by the researcher to lure other bots. Honeypots show non-human behaviors and can be made in groups to increase their effect by connecting and interacting with each other without intervention in activities of normal users. Due to the fact that they are designed in a way that any normal user can immediately tell they are bots, any user in the network that connects to a honeypot will be considered as a bot. By using this method, a researcher can gather a large set of bots active in a network with high confidence. This method has been applied [12], [20].

When building a classifier, the proposed method should be evaluated by means of an evaluation metric. The most widely used metrics in bot detection are accuracy and precision [3], [12], [16], [18], [20], [23]. Although they can reach extremely high precision, few report their recall. This drawback will have a significant effect as most of the bots will remain undetected (low recall) at the expense of precision.

III. Labeled Datasets for Bot Detection

Obtaining a social bot dataset can be difficult due to the challenge in obtaining definitive ground truth. While many approaches have been employed to obtain ground truth, we use two main approaches to label social media users as bots or humans: 1) observing whether the social media platform suspends these users and treating these suspended users as bots, and 2) creating "honeypot" bot accounts that to lure bots into following them, and then labeling these followers as bots.

In this work we create two datasets to test the bot detection approaches. The labels for each dataset are obtained using different labeling approaches. Here, we enumerate the two datasets used in this study, along with describing the bot labeling approach[4].

TABLE I: Statistics of the data used in this study.

Property	Libya	Arabic Honeypot
Tweets	1,150,192	504,679
Retweets	576,167	220,500
Unique Users	94,535	6,285
Labeling Approach	Suspended Accts	Honeypots
Bot Ratio	7.5%	49.0%

A. Arab Spring in Libya

From February 3^{rd}, 2011 to February 21^{st}, 2013, we collected Twitter data pertaining to Arab Spring activity in Libya. The data was collected from Twitter's Streaming API[5], a service which provides a stream of tweets matching a query [11]. The query we used to collect this data consisted of the following keywords: `#libya, #gaddafi, #benghazi, #brega, #misrata, #nalut, #nafusa, #rhaibat`, as well as a geographic bounding box around Libya[6]. Statistics on the dataset are shown in Table I.

We obtained labels of whether a user is a bot or human by observing how Twitter handled these users. In February of 2015, we crawled each user in the dataset and observed the status of his Twitter account. We observed the user's account status via the `statuses/user_timeline` API endpoint[7]. The status can take on one of three values:

1) **Active:** A user whose account is still open and available on the site.
2) **Deleted:** A user whose account has been deleted. A user can be deleted by violating Twitter's policies. This is considered a permanent ban.
3) **Suspended:** A user whose account has been suspended for violating Twitter's policies. This is considered a temporary ban, where the user can petition Twitter to have his account reinstated.

These labels were obtained by using Twitter's APIs to crawl the dataset and inspecting the response code from the API. Through this process we discovered that **92.5%** of the users are active, **4.7%** are deleted, and **2.8%** are suspended. Because deleted and suspended have similar meanings, we consider both labels as a bot.

At first glance, we notice that the fraction of users identified as bots by this labeling technique is around 7.5%. This distribution gives a very conservative estimate of the number of bots on Twitter. This is a side effect of an industry approach which focuses purely on precision in order to avoid accidentally deleting some real users. We realize that this is not representative of the true distribution of bots on the site [22],

[4]Both datasets can be obtained from the following link: http://www.public.asu.edu/~fmorstat/ASONAM_data.zip.

[5]https://dev.twitter.com/streaming/reference/post/statuses/filter

[6]Southwest Lng/Lat: 23.4/10.0; Northeast Lng/Lat: 33.0,25.0.

[7]https://dev.twitter.com/rest/reference/get/statuses/user_timeline

and that many bots may have been overlooked by Twitter. With this in mind, we introduce our next dataset which focuses on detecting bots in the wild through honeypots which tweet specific content.

B. Arabic Honeypot Dataset

This dataset consists of bots tweeting messages in Arabic. To collect this dataset, we construct a honeypot network. This network consists of 9 accounts controlled automatically by a single controller. Each account tweets messages containing Arabic phrases identified by a subject matter expert pertaining to a specific group of people. Each account also randomly follows other honeypots in our network. Since bots have a lower chance of forming social ties [19], we perform this random following process to lower the chance that our accounts are deleted by Twitter's automatic account removal algorithm. Additionally, each honeypot can randomly retweet one of the other honeypots it follows in order to give that honeypot prominence on the network and lower its probability of being deleted due to Twitter's policies.

All of the bots in our honeypot dataset behave identically, as dictated by the controller. Each bot collects tweets using the selected Arabic phrases from Twitter's Streaming API. At random time intervals, the bot either copies the tweet, passing it off as its own, or retweets it from the user who originally posted it. The full logic for the honeypot controller is shown in Algorithm 1. Using a network of 9 honeypots, we consider all followers of our honeypots to be bots: due to the way our honeypots behave, no normal user would follow them [12]. With this approach we collected 3,602 active bot accounts who followed one of our honeypot accounts.

```
while True do
    Randomly choose one honeypot, h;
    r ⟵ random number ∈ [1, 10];
    if r = 1 then
        h retweets the most recent tweet from another
          randomly-selected honeypot;
    end
    else
        Sample tweets from Twitter Streaming API for
          20 seconds, filtering based on Arabic phrases,
          call sample set S;
        r ⟵ random number ∈ [1, 10];
        s ⟵ randomly-selected tweet from S;
        if r < 3 then
            h retweets s;
        end
        else
            h copies s and tweets it word-for-word;
        end
    end
    Wait 10 seconds;
end
```

Algorithm 1: Logic for honeypot controller. The controller manages the honeypot accounts used to collect bots in the wild. All randomly-generated numbers mentioned in the pseudocode are generated uniformly at random.

While the honeypot method yields a set of bot accounts, it does not give us a set of real users. This is because we only look at the bot followers to our honeypots, and we do not have ground truth labels for real users. In order to test our model, we need to collect a set of real users to use as negative training instances. To do this, we manually inspected a seed set of 10 users who also tweeted the same phrases[8]. We did this to ensure that the algorithm we train finds bot patterns in the text, and does not simply learn the difference in language distributions. Moving forward with the assumption that real users do not follow malicious bots, we use 1-link snowball sampling to collect their immediate network. We ensured that each user in the sample had fewer than 1,000 followers to make sure that we did not collect any celebrities [24]. We also inspected each user in the sample to ensure that he tweeted one of the phrases at some point in his last 200 tweets. This approach yielded 3,107 real-world accounts, which helped us to maintain the same class distribution reported in other approaches [12].

IV. Bot Detection Approaches

We propose a model for bot detection which considers recall in its formulation. As baselines we consider popular heuristics used to detect bot accounts. We also consider a model based upon topic modeling. In this section we present the formulation for all of the bot-detection approaches we study in this work. We begin by introducing the heuristics and baselines considered in this work. We then continue to introduce our model, $BoostOR$.

A. Heuristics

Here we introduce a set of heuristics that are used to differentiate bot from non-bot users in social media. These heuristics are based upon state-of-the-art studies in bot detection on social media.

1) Fraction of Retweets: This measures the number of times the user published a retweet divided by the number of tweets the user has published, calculated as:

$$Heuristic_{Retweet}(u) = \frac{|\{x|x \in tweets^u, x \text{ is retweet}\}|}{|tweets^u|}. \tag{2}$$

Whether a tweet is a retweet is determined by looking at the "retweeted_status" field of the tweet's data when returned through the API. If the field contains tweet information, we consider it to be a retweet. This measure was introduced in [16], and hypothesizes that bots are unable to produce original content, so they rely on the retweet feature in Twitter in order to establish their presence.

2) Average Tweet Length: A tweet's text is limited to 140 characters, but it is possible that bots post fewer than that as they could just be promoting a URL or a hashtag [13]. To account for this, we introduce this heuristic which measures the average length of the user's tweets. It is the sum of the characters in all of the tweets the user has published divided by the number of tweets the user has published, formally:

$$Heuristic_{Length}(u) = \frac{\sum_i^{|tweets^u|} |tweets_i^u|}{|tweets^u|}, \tag{3}$$

[8]We did not extract verified accounts as these users are not normal users. They are often controlled by public relations firms and tweet on specific topics.

where $|tweets_i^u|$ is the length of user u's i-th tweet, measured by the number of characters in the tweet.

3) Fraction of URLs: This measures the number of times the user published a tweet containing a URL divided by the number of tweets the user has published, formally:

$$Heuristic_{URL}(u) = \frac{|\{x|x \in tweets^u, x \text{ contains URL}\}|}{|tweets^u|}. \quad (4)$$

Whether a tweet contains a URL is determined by looking at the "entities" field of the tweet returned by Twitter's API. This measure has been studied previously [15], [23], and hypothesizes that bots are motivated to persuade real users into visiting external sites operated by their controller. In this way, this heuristic traps bots that are trying to promote URLs in their tweets.

4) Average Time Between Tweets: It has been discovered that many bots tweet in a "bursty" nature [13], [23], publishing many of their tweets within a short amount of time. This behavior is measured as:

$$Heuristic_{Time}(u) = \frac{1}{|tweets^u| - 1} \sum_{i=2}^{N} (t_i - t_{i-1}), \quad (5)$$

where t_i is the time stamp of the i-th tweet in u's timeline, sorted chronologically in ascending order (i.e. $t_i \geq t_{i-1}$).

B. Topic Modeling

Bot accounts are created by malicious users to spread misinformation through their tweets. Thus, their content can be a strong indicator to expose such potential compromised accounts. The problem with using content for bot detection is that the raw text features are of high dimensionality and sparse, causing the "Curse of Dimensionality". Inspired by the recent advances of topic modeling, we adopt latent Dirichlet allocation (LDA) to obtain a topic representation of each user. Since the raw features (words) can be viewed as being generated by a mixture of topics, and the bots are naturally more interested in certain topics, denoting each user as a distribution over different topics may help the learning process better identify them from regular accounts.

LDA is an unsupervised and probabilistic model that has been proven useful for extracting latent semantics of documents. The principle idea behind LDA is that it treats each document as a distribution over topics, and each topic as a distribution over the vocabulary in the dataset. LDA requires one parameter[9], K, the number of topics in the corpus. From here, LDA learns two matrices:

1) Φ: the $Topic \times Word$ matrix. Each topic in LDA, Φ_i, is a probability distribution over the entire vocabulary in the corpus. Thus, Φ_i^j is the probability of word j occurring in topic i.
2) Θ: the $Document \times Topic$ matrix. Since each document is modeled as a distribution over topics, this each row, Θ_i, contains the document's distribution over all of the topics learned by LDA.

In this approach, we treat the user as a document. Each user's document consists of the concatenation of all of the content of his tweets. We feed these documents into LDA with $K = 200$, obtaining Φ and Θ values for the corpus[10]. Since the Θ matrix contains the affinity for each user to each topic, we can treat these as features to be fed into a classifier. We build an SVM classifier by directly using Θ as the $Instance \times Feature$ matrix, where the instances are users and the features are their affinities for the latent dimensions discovered by LDA.

C. Supervised Model for Bot Identification

In this section, we will introduce how we optimize F_1, which includes both precision and recall, when exposing bots. Since bots are usually generated by different parties for different purposes, the discriminant characteristics of bots from different groups are unrelated. Such heterogeneity of bots makes it challenging to come up with a classifier. To this end, we formulate the problem as a boosting task. Boosting methods aim to achieve an optimal classifier through ensembling weak classifiers, which are more proper for this problem since different weak classifiers will focus on different bots.

First, we investigate whether boosting algorithms are directly applicable for bot detection. For generality, we selected AdaBoost for optimization. AdaBoost trains one weak classifier based on all training examples every iteration. In each iteration, the misclassified examples in the previous round are higher weighted in the next round. The final classifier is the weighted ensemble of all weak classifiers. Therefore, the key issues are how the weight are determined for different weak classifiers and training examples. Here we use α to denote classifier weight, the weight of classifier t can be calculated as follows:

$$\alpha_t = -\frac{1}{2} ln(\beta_t), \quad (6)$$

where β_t denotes the extent of the base learner deviates from the optimal solution. β_t can be directly calculated with training error ϵ as follows:

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}. \quad (7)$$

As shown in Eq. 8, α_t is greater than zero if ϵ_t is less than 50%; while α_t is less than zero if ϵ_t is greater than 50%. ϵ_t can be calculated through Eq. 8.

$$\epsilon_t = \frac{1}{\sum_{i=1}^{m} \mathbf{D}_t(i)} \sum_{i=1}^{m} \mathbf{D}_t(i) \mathbf{1}(h_i(\mathbf{v}_i) \neq y_i), \quad (8)$$

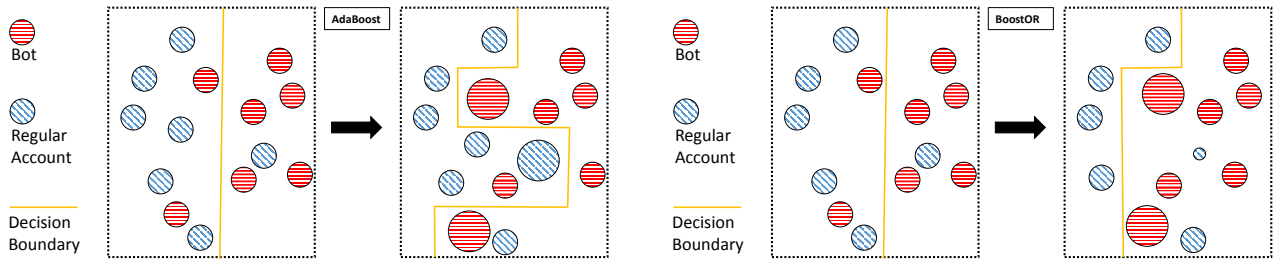
where the function $\mathbf{1}(\cdot)$ equals one when the condition holds, and zero otherwise.

The second issue is to determine the weight of each training instance at each iteration. The instance weight is regulated depending on whether it is correctly classified in the previous round and the performance of the weak learner. As denoted in Eq. 9, when the weak learner's error rate is less than 50%, weights of misclassified users will increase while those of the rest decrease.

$$\begin{aligned} \mathbf{D}_{t+1}(i) &= \frac{\mathbf{D}_t(i) exp(-\alpha_t y_i h_t(\mathbf{v}_i))}{Z_t} \\ Z_t &= \sum_{i=1}^{m} \mathbf{D}_t(i) exp(-\alpha_t y_i h_t(\mathbf{v}_i)). \end{aligned} \quad (9)$$

[9]We use the default hyperparameter value of $\alpha = \frac{1}{K}$, and $\beta = \frac{1}{K}$.

[10]We will discuss our selection of K in the experiments section.



(a) An illustrative example of AdaBoost.

(b) An illustrative example of BoostOR.

Fig. 1: Illustration of the models used in this work.

The algorithm of AdaBoost for bot detection is illustrated in Algorithm 2. Note that m is the number of all users and k is the number of features. Since LDA is adopted to represent the user, here each user vector $\mathbf{v}_i$ can be viewed as a probability distribution over k topics. The user labels are denoted by $\mathbf{Y} = \{y_1, \ldots, y_m\} \in \{-1, 1\}^{m\times 1}$, where $y_i = 1$ means that user i is a bot.

Input: The user-feature matrix $\mathbf{V} \in \mathbb{R}^{m\times k}$,
the user-label matrix $\mathbf{Y} \in \{-1, 1\}^{m\times 1}$ and a weak learner $h : \mathbf{V} \to \mathbf{Y}$
the initial user weight vector $\mathbf{D}_1 = (\mathbf{D}_1(1), \ldots, \mathbf{D}_1(m))$
the maximum number of iterations max_{iter}.
Output: The ensemble classifier
$H(\mathbf{v}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{v}))$
For t = 1, ..., max_{iter}
Train weak learner using instance weight $\mathbf{D}_t$;
Get the trained classifier $h_t : \mathbf{V} \to \mathbf{Y}$ and training error ϵ_t as Eq. 8;
Calculate the weight $\alpha_t \in \mathbb{R}$ for h_t as Eq. 6;
Update user weight as Eq. 9;
until Convergence.

Algorithm 2: AdaBoost for Bot Detection

Since we aim to achieve a classifier which is sensitive to bots, we investigate how AdaBoost could be adapted for optimizing recall. As mentioned before, the weight of bots may be reduced if they are correctly classified. The reduction of weight leads subsequent weak classifiers to less focus on these bots, which is unfavourable. Therefore, we next investigate how the sensitivity to bots can be kept through regulating the weights of training instances. An intuitive solution is to avoid reducing weights of bots, which can be formulated as follows:

$$\mathbf{D}_{t+1}(i) = \mathbf{D}_t(i)\beta^{-y_i|h_i(\mathbf{v}_i - y_i)|}. \tag{10}$$

As shown in Eq. 10, if a regular user is predicted to be a bot, the corresponding weight will be multiplied by $\beta^{|h_i(\mathbf{v}_i - y_i)|} \in (0, 1]$, while the mislabeled bots still gain more weights. The loss between prediction and ground truth spans between the range of $[0, 1]$, and an exponential form ensemble predictor is adopted, which both enable Theorem 1 to hold. This means that the training error after T iterations is bounded. We name the new boosting algorithm as Boosting through Optimizing Recall ($BoostOR$). The detailed algorithm is shown in Algorithm 2. Figure 1(a) illustrates how the example weight is updated in AdaBoost model. The mislabeled instances will gain a larger weight in the second round of iteration. While in $BoostOR$, the weight change depends on the labels of the user. If a bot is wrongly predicted, its weight is enlarged. mislabeled regular users are more often ignored.

However, two questions need to be answered about BoostOR: 1) will it converge by keeping weights of bots? 2) will trivial solutions be achieved by classifying all examples as bots? To answer the first question, we introduce Theorem 1 to indicate the convergence rate of the algorithm:

Theorem 1. *After T iterations, the average training loss of each iteration of BoostOR is upper bounded by:*

$$\begin{aligned} \min_{\mathbf{i}} \quad & \frac{L_i}{T} + \frac{\sqrt{2\hat{L}\ln(m)}}{T} + \frac{\ln(m)}{T} \\ where \quad & L_i = \frac{1}{2}\sum_{j=1}^{T} |h_j(\mathbf{v}_i - y_i)| \end{aligned} \tag{11}$$

The corresponding proof can be found in [5], where the range of convergence rate is also provided. The rate of convergence is no more than $O(\sqrt{\ln(m)/T})$ and can be as fast as $O(\ln(m)/T)$. $\hat{L}$ is the loss of optimal solution.

Since the weight of regular users are reduced instead of removed, trivial solutions will not be easily achieved in real world data, where bots are the minority group. In next section, we empirically prove this with two real world Twitter datasets.

Input: The user-feature matrix $\mathbf{V} \in \mathbb{R}^{m\times k}$,
the user-label matrix $\mathbf{Y} \in \{1, -1\}^{m\times 1}$ and a weak learner $h : \mathbf{v} \to y$
the initial user weight vector $\mathbf{D}_1 = (\mathbf{D}_1(1), \ldots, \mathbf{D}_1(m))$
the maximum number of iterations max_{iter}.
Output: The ensemble classifier
$H(\mathbf{v}) = sign(\Pi_{i=1}^{max_{iter}}(\beta_i^{-h_i(\mathbf{v})} - \beta_i^{-\frac{1}{2}})$
For t = 1, ..., max_{iter}
Train weak learner using instance weight $\mathbf{D}_t$;
Get the trained classifier $h_t : \mathbf{V} \to \mathbf{Y}$ and training error ϵ_t as Eq. 8;
Calculate the weight $\alpha_t \in \mathbb{R}$ for h_t as Eq. 6;
Update user weight as Eq. 10;
until Convergence.

Algorithm 3: BoostOR for Bot Detection

TABLE II: Confusion matrix for the $Heuristic_{Time}$ measure on the Arabic Honeypot Dataset.

		Prediction	
		Bot	Human
Truth	Bot	49.9%	50%
	Human	0%	0.1%

TABLE III: Confusion matrix for the $BoostOR$ measure on the Arabic Honeypot Dataset.

		Prediction	
		Bot	Human
Truth	Bot	42.8%	7.7%
	Human	14.8%	34.7%

V. Experiments

In this section we empirically investigate the performance of *BoostOR* with respect the ability to maximize the F_1 score on bot detection. We perform two experiments: first to show the performance of different algorithms, and second to show how the parameters used in building the AdaBoost and BoostOR models can influence the results of these methods.

A. Bot Classification

We apply each heuristic, as well as AdaBoost and BoostOR to both the Libya dataset, shown in Table IV, and the Arabic Honeypot dataset, shown in Table V. We find that both AdaBoost and BoostOR outperform the heuristics, with BoostOR performing the best on both datasets. Interestingly, we find that SVM underperforms, achieving a worse result than the heuristics when optimizing the F_1 score.

One thing to note about the heuristics is that they conform to the class distribution. That is, when they achieve their maximum performance they are simply *always* predicting the user is a bot. In other words, the results of heuristic measures indicate that we should delete our entire dataset for both datasets. While this yields what seem to be competitive results, the implication of this approach is not reasonable. While the F_1 score is useful to measure the performance, we need to dig deeper to understand the implication of these results.

To illustrate the difference in performance between the heuristics and our proposed model, we show a confusion matrix of the best-performing heuristic, $Heuristic_{Time}$, in Table II and a confusion matrix of $BoostOR$ in Table III on the Arabic Honeypot dataset. First, we notice that the heuristic *never* misclassifies as a human as a bot. This is in line with the goals of the industry. However, we see in the $BoostOR$ results in Table III that we achieve superior performance by lowering the false-negative rate of the model, which is in line with the goals of the researcher.

B. Sensitivity of Topic Modeling

In previous experiments, the number of topics, K, of LDA is set as 200 for all methods. In this section, we study how the number of topics will influence the performance of the proposed model, BoostOR. The change of precision, recall and F_1 score as a function of K is illustrated in Figure 2. On both datasets, the dimensionality spans from 50 to 500 and the variations of performance are observed.

As shown in the figures, the best performance is achieved when there are around 200 topics. When K deviates from the optimal value of 200, no matter increasing or decreasing, the performance decreases. Too many topics enable some redundant and meaningless topics to exist, while some important topics may be neglected if the K is too small.

TABLE IV: The Precision, Recall and F_1 measure of different models on Libya dataset.

Method	Precision	Recall	F_1
$Heuristic_{URL}$	6.74%	65.12%	12.21%
$Heuristic_{Retweet\%}$	7.73%	53.63%	13.51%
$Heuristic_{Length}$	7.74%	53.63%	13.51%
$Heuristic_{Time}$	7.48%	99.89%	13.91%
SVM	29.24%	8.78%	13.53%
$AdaBoost$	75.25%	7.48%	13.61%
$BoostOR$	75.41%	8.14%	**14.69%**

TABLE V: The Precision, Recall and F_1 measure of different models on Honeypot dataset.

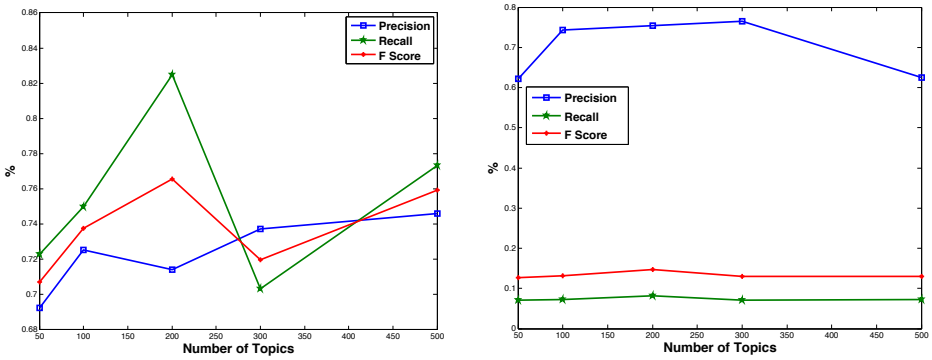
Method	Precision	Recall	F_1
$Heuristic_{URL}$	49.69%	96.39%	65.58%
$Heuristic_{Retweet\%}$	50.05%	99.33%	66.56%
$Heuristic_{Length}$	50.00%	99.82%	66.63%
$Heuristic_{Time}$	49.99%	99.96%	66.65%
SVM	62.41%	62.52%	62.47%
$AdaBoost$	79.76%	72.41%	75.91%
$BoostOR$	71.42%	82.48%	**76.55%**

VI. Conclusion

In this work we explore the problem of finding bots on social media. We begin by collecting two datasets, each with different labeling mechanisms. Then we continue to propose a bot detection method that optimizes the F_1 score of the model, which considers recall in addition to precision.

The datasets we use to test this method were labeled through different processes. The first dataset we consider in this work is labeled through using Twitter's labeling processes of active/suspended/deleted users. The second dataset was obtained by building a honeypot network and collecting the users who follow our honeypot accounts. These datasets encompass different topics and were labeled using two different methods, which make them ideal for testing both the performance and generalizability of our model.

Next, we propose to optimize the F_1 score of the bot detection problem through boosting a basic learner. In order to recall more bots from the dataset, the proposed BoostOR focuses more on the mislabeled bots and downweights mislabeled regular users. The theoretical analysis shows that the optimization can be done efficiently. Experimental results on two real world datasets show that our model outperforms the



(a) Honeypot dataset (b) Libya dataset

Fig. 2: Precision, recall, and F_1 score of BoostOR with varying number of topics.

state-of-the-art bot detection techniques as well as heuristics which are often considered for this problem.

Future work seeks to further improve the recall of the bot detection task. Existing approaches which focus on precision could potentially be modified to improve recall. We wish to exploit the sparsity of text in order to better separate bots from normal users. Additionally, we will continue to refine the process for collecting labeled social media datasets.

ACKNOWLEDGMENT

Support was provided, in part, by the Office of Naval Research through MINERVA N000141310835 on State Stability.

The authors would like to thank Wei Wei for sharing his active/suspended user labels for the Libya dataset.

REFERENCES

[1] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network: when bots socialize for fame and money. In *Annual Computer Security Applications Conference*, pages 93–102. ACM, 2011.

[2] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. Design and analysis of a social botnet. *Computer Networks*, 57(2):556–578, 2013.

[3] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Who is tweeting on twitter: human, bot, or cyborg? In *Annual Computer Security Applications Conference*, pages 21–30. ACM, 2010.

[4] David M Cook, Benjamin Waugh, Maldini Abdipanah, Omid Hashemi, and Shaquille Abdul Rahman. Twitter Deception and Influence: Issues of Identity, Slacktivism, and Puppetry. *Journal of Information Warfare*, 13(1):58–71, 2014.

[5] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[6] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @spam: the underground on 140 characters or less. In *Conference on Computer and Communications Security*, pages 27–37. ACM, 2010.

[7] John P. John, Alexander Moshchuk, Steven D. Gribble, and Arvind Krishnamurthy. Studying Spamming Botnets Using Botlab. In *Networked Systems Design and Implementation*, volume 9, pages 291–306, 2009.

[8] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *Conference on Computer and Communications Security*, pages 3–14. ACM, 2008.

[9] Yoree Koh. Only 11% of new twitter users in 2012 are still tweeting. http://blogs.wsj.com/digits/2014/03/21/new-report-spotlights-twitters-retention-problem/. Accessed: 2015-03-31.

[10] Brian Krebs. Twitter Bots Drown Out Anti-Kremlin Tweets. http://krebsonsecurity.com/2011/12/twitter-bots-drown-out-anti-kremlin-tweets/, 2011.

[11] Shamanth Kumar, Fred Morstatter, and Huan Liu. *Twitter Data Analytics*. Springer, 2014.

[12] Kyumin Lee, Brian David Eoff, and James Caverlee. Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. In *ICWSM*. AAAI, 2011.

[13] Sangho Lee and Jong Kim. Early filtering of ephemeral malicious accounts on Twitter. *Computer Communications*, 54:48–57, 2014.

[14] Yelena Mejova, Ingmar Weber, and Michael W Macy. *Twitter: A Digital Socioscope*. Cambridge University Press, 2015.

[15] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. Truthy: mapping the spread of astroturf in microblog streams. In *World Wide Web Companion*, pages 249–252. ACM, 2011.

[16] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonalves, Alessandro Flammini, , and Filippo Menczer. Detecting and Tracking Political Abuse in Social Media. In *ICWSM*, pages 297–304. AAAI, 2011.

[17] Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, and Giovanni Vigna. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2011.

[18] Kurt Thomas, Chris Grier, and Vern Paxson. Adapting social spam infrastructure for political censorship. In *Conference on Large-Scale Exploits and Emergent Threats*. USENIX, 2012.

[19] Kurt Thomas, Chris Grier, Dawn Song, and Vern Paxson. Suspended accounts in retrospect: an analysis of twitter spam. In *Internet Measurement Conference*, pages 243–258. ACM, 2011.

[20] Olivier Thonnard and Marc Dacier. A strategic analysis of spam botnets operations. In *Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, pages 162–171. ACM, 2011.

[21] Alex Hai Wang. Detecting spam bots in online social networking sites: a machine learning approach. In *Data and Applications Security and Privacy XXIV*, pages 335–342. Springer, 2010.

[22] Wei Wei, Kenneth Joseph, Huan Liu, and Kathleen M Carley. The fragility of Twitter social networks against suspended users. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 9–16. IEEE, 2015.

[23] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, 38(4):171–182, 2008.

[24] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. *Social Media Mining: An Introduction*. Cambridge University Press, 2014.

[25] Reza Zafarani and Huan Liu. 10 bits of surprise: Detecting malicious users with minimum information. In *Conference on Information and Knowledge Management*, pages 423–431. ACM, 2015.