BotWalk: Efficient Adaptive Exploration of Twitter Bot Networks

Amanda Minnich*†, Nikan Chavoshi*, Danai Koutra‡, and Abdullah Mueen*

*University of New Mexico ‡University of Michigan

Abstract—**We propose BotWalk, a near-real time adaptive Twitter exploration algorithm to identify bots exhibiting novel behavior. Due to suspension pressure, Twitter bots are constantly changing their behavior to evade detection. Traditional supervised approaches to bot detection are non-adaptive and thus cannot identify novel bot behaviors. We therefore devise an *unsupervised* approach, which allows us to identify bots as they evolve. We characterize users with a behavioral feature vector which consists of (well-studied in isolation) metadata-, content-, temporal-, and network-based features. We identify a random bot from our seed bank, populated initially by previously-labeled bots, gather this user's followers' features from Twitter in real time, and employ an unsupervised ensemble anomaly detection method in the multi-dimensional behavioral space. These potential bots are folded into the seed bank and the process is then repeated, with the new seeds' features allowing us to adaptively identify novel bot behavior. BotWalk allows for the identification of on average 6,000 potential bots a day. Our method allowed us to detect 7,995 previously undiscovered bots from a sample of 15 seed bots with a precision of 90%.**

I. Introduction

More than a billion people use online social networks for information dissemination, entertainment, and social interaction. Unfortunately, these platforms are exploited by abusive automated accounts, also known as bots, for financial or political gain. The presence of these bots is a constantly growing problem that compromises the empowerment of online social communities. Automated accounts easily allow botmasters to spam inappropriate content [1], participate in sponsored activities [2], and make money by selling accounts with human followers [3]. An estimated 8.5% of Twitter accounts are bots [4] and the bots are growing at a higher rate than the rate at which Twitter suspends them.

The identification and suspension of bots is a challenging problem for several reasons: these social networks are massive – Twitter alone is estimated to contain over 300 million users and billions of edges. Furthermore, the amortized cost of creating a bot[1] is much less than that of detecting a bot, and the cost of suspending a user incorrectly is much higher. Lastly, the bot-masters creating these automated accounts are constantly evolving the bots' behavior to attempt to evade detection. This is the typical 'Red Queen' effect, well known in biology and cybersecurity, in which "It takes all the running you can do, to keep in the same place"[5]. As Twitter's detection methods evolve, so do the botmasters'. To win this arms race, or to even keep up, a bot detection method must have the following characteristics:

- Very high *detection rate*, ideally higher than the rate at which automated accounts can be created, which would guarantee eradication of bots.
- *Scalable*, the method is robust in the face of increasing social network size.
- *Adaptive*, which means it retains the above two properties when bot-masters evolve. Simply training on labeled data that matches the current state is not enough, for as the bots' behaviors change, static classifiers will quickly become obsolete. This indicates that *unsupervised* solutions that detect novel, anomalous behavior are likely to be more effective in this quickly evolving environment.
- Completely *automated*, so that it can keep up with the rate of bot-master activity.

In this paper, we propose a bot detection technique using Twitter's restricted API access for online updates that is adaptive, scalable and has the highest detection rate of current methods. Our BotWalk algorithm uses an adaptive search strategy to maximize detection rate in a rapidly changing social network. In the Twitter network, BotWalk can identify up to 6000 bots-per-day and adapt to detect novel bot behaviors automatically.

Problem Formulation: The high-level goal of this work is to efficiently explore the evolving Twitter network and identify bots manifesting continually changing behavioral patterns.

Contributions:

- Creation of BotWalk, a near-real time adaptive anomaly detection framework with 90% precision in detecting bot behavior in Twitter data.
- Implementation of an adaptive approach to feature selection, necessitated by the limited amount of data accessible in real-time and the rapidly changing Twitter environment.
- Utilization of domain knowledge to intelligently partition the feature space, leading to up to a 30% increase in precision.
- Assembly of a comprehensive Twitter dataset, collected over the course of formulating this work, and made available to the public [6].

II. Related Work

Twitter bot detection has received significant interest in the literature. Most current approaches focus on supervised, non-adaptive methods [7][8][9]. Unfortunately, a supervised approach has several shortcomings. The classification is only as good as the labeled data, which is often biased or outdated. Supervised algorithms can be useful when trying to classify

†`aminnich@cs.unm.edu`

[1]Cost of creating a bot is equivalent to a mouse click by a human to solve re-captcha.

a given user as a bot, but what if the goal is identification of new bots? And what if these bots are constantly exhibiting new behavioral patterns? It is in these cases that supervised algorithms fall short.

In [10], the authors present an approach to semi-automatically label users as bots by identifying several features that can discriminate between obvious bots and human users. For each feature, the threshold value has to be manually set, and these values can vary depending on the candidate bot set, which reduces the automated aspect of the work. Other feature-based approaches use only a few linguistic attributes and test on a small number of manually-labeled accounts [11], or only temporal features in a supervised [12] or unsupervised way by identifying highly-correlated activity streams [13]. The latter approach has high recall and a low false positive rate, but is very easy to evade since it is just based on one feature. One study extracted a large number of features [14], but they used them in a supervised fashion. We compare their detection rate to BotWalk's in Section VI-C.4.

Our work is unique because it is an unsupervised exploratory method that is able to adaptively identify novel bot behavior using a variety of features which capture different dimensions of behavioral profiles. By using an unsupervised method seeded from known bots, rather than simply training a classifier on labeled data, BotWalk is able to discover new, unknown behavioral patterns.

III. Framework

Bot detection in a rapidly changing social network with limited external visibility is a challenging computational problem. As with malware, bot masters continually modify bot characteristics including communication patterns, content, and connectivity. Additionally, Twitter's query interface restrictions place significant limitations on the amount of new data that can be collected relative to the amount being generated.

Figure 1 shows our approach, which combines known data, domain expertise, and unsupervised anomaly detection. As described in Algorithm 1, we begin by selecting a small set of seed bots (see Section VI-B) and a set of random users. This random set can consist of both normal users and bots, as it is not labeled. However, since estimates say that at most 8.5% of Twitter users are bots [4], a random sample should on average adhere to this constraint. The anomaly detection algorithm considers the random user set as containing the 'normal' group.

For each Twitter user from the seed bot neighborhood and random user sets, metadata such as timeline and user account information is collected (Lines 6 – 10). Features of the seed bot neighborhood and random user sets are then assembled from these raw data (Lines 11 – 13). Feature selection and assembly is a major contribution of this work, and is discussed in Section IV.

Finally, the ensemble anomaly detection algorithm is applied, resulting in identification of bots and normal users. Elements from these sets are used to update the seed bot

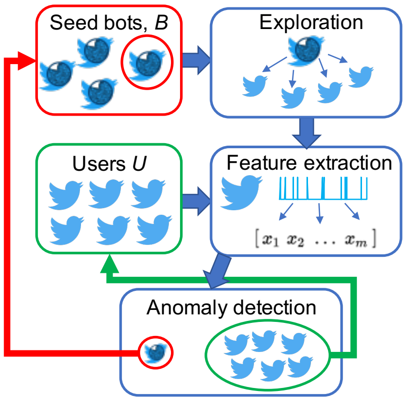


Fig. 1: Overall framework of our bot identification algorithm

and random user sets respectively (Lines 14 – 16). The anomaly detection algorithms and our partitioning approach to ensemble anomaly detection is the topic of Section V.

Algorithm 1 NetworkExploration

```
1:  Input: Set of seed bots $\mathcal{B}$, Set of random users $\mathcal{U}$;
2:  while $explore = True$ do
3:      $b$ = pop random element from $\mathcal{B}$;
4:      $\mathbf{F_{user}}$ = feature matrix of $\mathcal{U}$;
5:      $N(b)$ = follower neighborhood of $b$;
6:      for neighbor $c$ in $N(b)$ do
7:          // Metadata via the Twitter REST API
8:          $I_M(c) = \texttt{get\_profile\_info(c)};$
9:          // Timeline info: 200 most recent tweets
10:         $I_T(c) = \texttt{get\_timeline\_info(c)};$
11:         // Extract metadata-, content-, network-, and
12:         // temporal-based features (Secs. IV-A-IV-D)
13:         $\mathbf{F_N}(c) = \texttt{create\_feat\_vector}(I_M(c), I_T(c));$
14:     $(\mathcal{B}_{out}, \mathcal{U}_{out}) = \texttt{UnsupervisedOutlierDet(}$
        $[\mathbf{F_{user}},\ \mathbf{F_N}]$); // (Secs. IV-A)
15:     $\mathcal{B}.update(\mathcal{B}_{out})$;
16:     $\mathcal{U}.update(\mathcal{U}_{out})$;
```

Our exploration method combines depth first exploration and breadth first exploration. Depth-first exploration constrains the search space but limits the scope, so if a human node is reached it can be difficult to return to a bot node. Breadth-first exploration maximizes the likelihood of reaching other bots at each step, since it is collecting all the followers of the current anomalous node, but the scale quickly explodes (for example, some Twitter users have tens of thousands of followers). Our method allows us to maximize the likelihood of identifying bots by collecting and analyzing the followers of the current seed bot, while still constraining the search space by continuously repeating the exploration process starting from individual random anomalous users. This has the added benefit of making our approach scalable by construction; by using a limited, carefully-chosen stream selected from this huge, constantly-changing network, this method is robust in the presence of increasing network size.

	Meta-data	Content	Network	Temporal	Total
Before/ After feature selection	9,035/ 127	23/ 22	5/ 5	14/ 14	9,077/ 168

TABLE I: Description of feature set

IV. Feature Selection and Data Collection

To generate a comprehensive behavior profile for a given user, we compile a large collection of features which capture different aspects of a user's behavior. Each feature (e.g., temporal bursts) alone has been shown to be effective for identifying bots in a *supervised* setting (note that we target unsupervised settings). However, bots are a diverse group and can have many different behavioral patterns. Just looking for bots with spamming content, for example, could ignore bots that are trying to farm followers. By combining a large collection of features indicative of different aspects of behavior, we are reducing bias and expanding the quantity and types of bots we can identify.

There is considerable interest in the machine learning community in using latent features or learned representations, rather than engineered features. While this has been shown to be effective in some cases (such as classification and link prediction), it requires a huge learning space. Online bot behavior changes rapidly due to suspension pressure [5]. With severely rate-limited data acquisition, massive data updates are not feasible and impede latent feature generation as the dataset quickly becomes stale. Additionally, in the case of identifying bots and potentially suspending users, there needs to be a level of interpretability in the results. By using features that can be understood, rather than latent features, we address this need.

Our features can be divided into four categories: metadata-, content-, temporal-, and network-based. We will describe a subset of members of each category, as well as the intuition behind them. A full listing of the features from each category will be available on the website accompanying this paper [6].

A. Metadata-based features

Features that characterize a user's profile shed light on the level of effort a user put in when generating it. We would expect a bot to have this process automated, so there may be parts missing or repeated [15]. Bots may also want to provide less information because what they are claiming is false, e.g., if the account claims the user is from California but all of its tweets are coming from China, it would not want the geo-enabled setting to be turned on. Real users often like these types of features because they provide intelligent shortcuts when entering content. Other attributes like age of the account and whether the username was auto-generated can provide valuable information about the likelihood that an account is a bot. In addition to these features, we also look at whether an account is protected or verified, neither of which a bot account is likely to be, their overall number of statuses and number of followers, both of which are likely to be higher in a bot account that has managed to persist on Twitter, and several other metadata-related features.

B. Content-based features

Automated accounts are created for a specific purpose. Whether it is to gain followers for marketing campaigns, disseminate information, spam sponsored content, or learn about other users, there is a goal in mind for each account. There are many features in the content of the tweets themselves that can capture this goal-oriented behavior. Looking at items like hashtags, URLs, and domains, both on a tweet and user level, has been shown to be effective in identifying bots [11][16]. Identifying repetition in these entities is also informative. For example, a bot trying to direct followers to a certain site will want to include that site's URL in as many tweets as possible. Perhaps they try to use different URL shorteners to camouflage this effort, so looking for repeated domains in the extended URL is also helpful [17]. For hashtags, URLs, and domains, we look at both the average number of entity per tweet as well as the average number of tweets with that entity. We also quantify the number of duplicate hashtags, URLs, and domains. Another informative feature is the normalized retweet count. Creating original content is costly for automated accounts, so retweeting is an easy way to add a life-like feel to a profile without having to generate this content [18]. Additional features we examine include the maximum, minimum, mean, and standard deviation of the Jaccard similarity of inter-tweet bags-of-words, the number of special characters, and tweet lengths.

C. Temporal-based features

Analyzing the time series of tweets is a powerful way to distinguish between bots and humans; indeed, whole papers have been written on bot identification based solely on temporal characteristics [12][13][19]. Because there are limits to how rapidly and how often a human being can tweet, quantification of burstiness, defined as $\frac{\sigma-\mu}{\sigma+\mu}$ [12], the average number of tweets per day, and the duration of the longest tweet session without a 10 minute break are informative features. Statistics describing the minimum, maximum, mean, standard deviation, and entropy of the inter-arrival time of consecutive tweets help to identify bots whose goal is to get as much content out as quickly as possible, or whose activity is on a programmed schedule. Other features we use to identify scheduled behavior are the p-values of the χ^2 test applied to the second-of-minute, minute-of-hour, and hour-of-day distributions, which evaluates whether tweets are drawn uniformly across these distributions.

D. Network-based features

Network-based features can be very helpful in characterizing Twitter user behavior. Research has shown that bots that persist in the Twitter network tend to amass a large number of followers [20] and friends. Figure 2 shows the connections between a set of previously-labeled bots.

There are a variety of connections that can join nodes, including follow, friends, and mention connections. We take

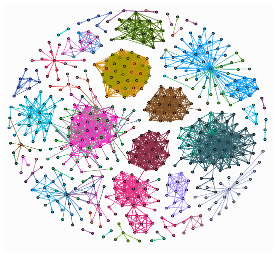


Fig. 2: Follower relationships between Twitter bots. Node colors represent highly-correlated activity stream clusters (see Section VI-B). Note the highly-connected nature of many of the bots.

Number of Nodes	Number of Edges	Number of complete out-edge sets	Number of 48hr streams collected
362,000	226 Million	110,800	75,000

TABLE II: Statistics of our publicly-available dataset

an ego-based approach to these relationships, quantifying the out-degree of the user's follow network, the out-degree of the user's friend network, and the out-degree of the user's mention network. These help to summarize the user's connectivity to others in a variety of ways.

E. Feature selection and normalization

After encoding categorical features using one-hot encoding we have over 9000 features. It is likely that not all of these feature are informative, so we choose to remove features that have the same value in 99.9% of samples, which reduces the feature space to 130 features. We then use the L2 norm to independently normalize each sample. For the partitioned outlier detection method, described in Section V-C, we perform feature selection and feature normalization on each subset separately.

F. Publicly-available dataset

Through the course of this research, we collected and analyzed a large body of Twitter data, which we are releasing for public use on our paper page [6]. This dataset was collected using the Twitter API and the Tweepy Python library [21]. We collected 5 types of information: friendship relationships, follower sets, activity streams, timelines (which includes a user's 200 most recent tweets), and user metadata. We store the bot and follower metadata in a PostgreSQL database (exported to CSV files for public release) and the user stream and timeline data in JSON files. Table II contains basic statistics of this dataset.

V. Ensemble Anomaly Detection

In order to adaptively identify bots with ever-changing behavior, we employ an unsupervised anomaly detection approach. Anomaly detection has been shown to be effective in identifying samples with previously-unseen behavior [22], so it is appropriate for this application. We employ true unsupervised anomaly detection in our method, which learns from the structure of the data itself with no outside guidance or labels.

In general, anomaly detection algorithms can be classified into four broad categories: density-based; distance-based; angle-based; and, more recently, isolation-based. To capture a variety of types of anomalous behavior, we use an ensemble of outlier detection algorithms, with one from each category. We choose to use Local Outlier Factor (LOF) as our baseline for comparison, since it is well-known, commonly used, and shown to be effective on a wide variety of data [23]. We find that this ensembling method improves precision by 5% when compared to LOF alone.

Algorithm 2 `UnsupervisedOutlierDet`

```
1:  Input: Feature Matrix F
2:  for column c in F do
3:      $\sigma_{\mathbf{c}}^2 = \frac{\sum_{i=1}^{N}(\mathbf{c}-\mu)^2}{N}$;
4:      if $\sigma_{\mathbf{c}}^2 < 0.001$ then
5:          Remove c from F;
6:  for row r in F do
7:      $\mathbf{x} = \frac{\mathbf{r}}{\sqrt{r_1^2+...+r_i^2+...+r_n^2}}$; // Normalize row
8:      Normalized LOF score, $s_{LOF} = Eq8(Eq2(\mathbf{x}))$;
9:      Normalized distance score, $s_D = Eq8(Eq4(\mathbf{x}))$;
10:     Normalized cos distance score, $s_C = Eq8(Eq5(\mathbf{x}))$;
11:     Normalized IF score, $s_{IF} = Eq8(Eq6(\mathbf{x}))$;
12:     $s_r = \frac{s_{LOF}+s_D+s_C+s_{IF}}{4}$;
13: Return: Scores S;
```

A. Anomaly Detection Algorithms

1) Local Outlier Factor: The density-based outlier detection algorithm that we user is Local Outlier Factor (LOF) [23]. LOF uses the notion of k-distance ($dist_k(x)$), which is defined as the distance to the k-th nearest neighbor of a point. Local reachability distance (lrd_k) of x is the average of the reachability distances from x's neighbors to x. Here $N_k(x)$ is the set of k-nearest neighbors of x.

$$lrd_k(x) = \frac{||N_k(x)||}{\sum_{y \in N_k(x)} \max(dist_k(y), dist(x, y))} \quad (1)$$

The LOF of a point x is the average of the ratios of the *local reachability* of x and its k-nearest neighbors. LOF can capture several normal clusters of arbitrary densities, which makes it robust for any data domain. Formally, LOF is defined as below:

$$LOF_k(x) = \frac{\sum_{y \in N_k(x)} \frac{lrd_k(y)}{lrd_k(x)}}{||N_k(x)||} \quad (2)$$

2) Distance- and Angle-Based Methods: In addition to a density-based measure, we use distance- and angle-based measures. We first generate an ideal 'normal' node by calculating the median of each feature.

$$\mathbf{c} = median(\mathbf{col}) \ \forall \ \mathbf{col} \ in \ \mathbf{F} \quad (3)$$

We then find the Euclidean distance between every user $\mathbf{x}$ and this ideal individual $\mathbf{c}$, giving us a distance-based outlier score.

$$DBD(\mathbf{x}) = \sqrt{(x_1 - c_1)^2 + ... + (x_n - c_n)^2} \quad (4)$$

To calculate an angle-based outlier score, we use the same center node but calculate the cosine distance between the two nodes.

$$ABD(\mathbf{x}) = \frac{\mathbf{x} \cdot \mathbf{c}}{||\mathbf{x}||\ ||\mathbf{c}||} \tag{5}$$

3) Isolation-based Method: Lastly, we use an Isolation Forest algorithm [24]. Rather than constructing an idea of 'normality' and identifying instances that differ from that, this algorithm instead 'isolates' outliers from normal samples. Given our feature matrix $\mathbf{F}$, this algorithm recursively splits the rows of $\mathbf{F}$ by randomly selecting a column $\mathbf{c}$ and a split value s, where $min(\mathbf{c}) <= s <= max(\mathbf{c})$. This recursive splitting forms a tree structure, and an Isolation Forest contains k such trees. The anomaly score is then defined based on the path length $h(\mathbf{x})$, which is the number of edges a sample traverses before terminating in an external node. Specifically, the anomaly score for a sample $\mathbf{x}$ from a size n dataset is defined by the equation:

$$s(\mathbf{x}, n) = 2^{-\frac{E(h(\mathbf{x}))}{c(n)}} \tag{6}$$

where $E(h(\mathbf{x}))$ is the average of $h(\mathbf{x})$ from a collection of isolation trees, and $c(n)$ is the average path length of an unsuccessful search in a Binary Search Tree, which is defined as

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{7}$$

Intuitively, if a sample has very anomalous feature values, it is going to be easily split from the remaining samples and thus will have, on average, a much shorter path length than a normal sample. Thus, when $E(h(\mathbf{x}))$ is close to 0, $s(\mathbf{x}, n)$ is close to 1, indicating it is very likely an anomalous sample. When $E(h(\mathbf{x}))$ is close to n, $s(\mathbf{x}, n)$ is close to 0, indicating it is a normal sample.

B. Combining different anomaly detection scores

One of the major challenges with creating a method based on an ensemble of anomaly detection algorithms is that it requires the combination of multiple scores that are not on the same scale. Equation 2 returns a local reachability score, Equation 4 returns a distance, Equation 5 returns a cosine distance, and Equation 6 returns a score based on an averaged path length. There are several different methods that can be used to combine these scores, depending on the potential application. Some methods focus on unifying rankings, while others focus on normalizing and combining scores [25]. For our application, having one unified score is preferable to having an overall ranking, thus we seek to normalize and combine these four scores. Our method is as follows, based on [26].

Let a user be $\mathbf{x}$ and the outlier score of $\mathbf{x}$ is $s(\mathbf{x})$. We scale each $s(\mathbf{x})$ using a Gaussian distribution to produce a probability $p(\mathbf{x})$ between 0 and 1 of the user $\mathbf{x}$ being an outlier.

$$p(\mathbf{x}) = \max(0, erf(\frac{s(\mathbf{x}) - \mu_s}{\sqrt{2}\sigma_s})) \tag{8}$$

We then can average these probabilities to produce one outlier score per user.

C. Applying domain knowledge to improve performance

In addition to human understanding, another advantage to using interpretable features is that we can employ domain knowledge of the feature space to improve performance. As described in Section IV, we have four feature categories: metadata-, content-, temporal-, and network-based. These feature subsets describe different aspects of a user's online behavior and have different feature spaces and scales. For example, the majority of the metadata features are categorical, e.g., language and time zone, and thus explode into a large feature space when using one-hot encoding. This could potentially overwhelm other features, leading to results biased towards metadata anomalies. Based on this observation, we choose to *partition* the feature space into these four subdomains and apply feature selection, sample normalization, and anomaly detection *separately* in each feature sub-space. We then combine these scores using the method described in the previous section. We perform this partitioned anomaly detection both with the ensemble of outlier detection algorithms and local outlier factor, as a baseline. Our experimental results show that this separation increases the precision of LOF by 30% and the ensemble method by 25%, achieving a precision of 90% for both methods.

VI. Experimental Analysis

A. Real-time data collection

Section III described the high-level exploration algorithm. We now go into the details of the actual execution. All code from these experiments will be released on GitHub [27].

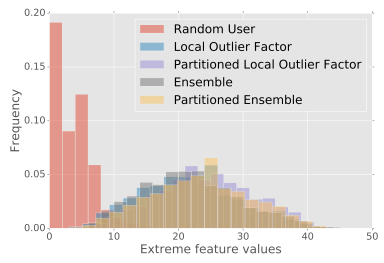
To expand from a seed user using the Twitter REST API [28], we first collect the follower set of this user, limited to the 5000 most recent followers to constrain the search space. (These followers are also less likely to be suspended since they have performed a recent activity, so in this way we avoid wasting queries.) We then collect the *timeline*, which is the 200 most recent tweets by the user, and the *user_information*, which is the metadata associated with this account, for every follower. We can collect 180 timelines every 15 minutes and metadata for 90,000 users every 15 minutes (queries must be performed sequentially, not concurrently). Once we have collected these data, we extract our comprehensive set of features and perform anomaly detection.

B. Experimental Design

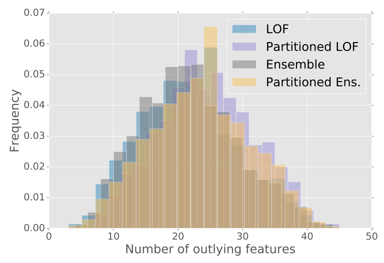
We design our experiments to assess the following items:

- Is our algorithm effective at identifying bots?
- As we explore and replace the seed bots, do we continue to find high-quality results?
- Are we able to identify bots with novel behavior as we explore?

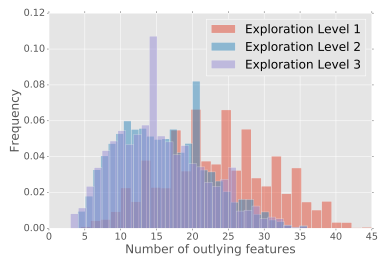
For our experiments, we first randomly select 15 non-suspended bots from our labeled dataset. This dataset contains $\sim 700,000$ labeled bot accounts identified by DeBot [13], which identifies users with highly-correlated activity streams. Debot has a mathematically proven false



(a) Distribution of number of features in the 10th or 90th percentile per user for 17,000 randomly-selected users versus BotWalk-detected outliers



(b) Zoomed in view of the distribution of the number of features in the 10th or 90th percentile per user for outliers detected in the 4 experiments.



(c) Comparison of of the distribution of the number of features in the 10th or 90th percentile per user for exploration Levels 1, 2, and 3

Fig. 3: (left) BotWalk-detected anomalous users have many more 'extreme' feature values when compared to random users; (center) Partitioned anomaly detection methods identify outliers with more 'extreme' feature values than those applied to the feature set as a whole; (right) Anomalies identified in the first round of exploration have a different extreme feature distribution when compared with those in the next two rounds, which are very similar.

positive rate of very close to zero [19], so we can have high confidence in the accuracy of these labels. Note that these users represent a specific subset of bot behavior, as they are so-called 'dumb' bots, with an obvious behavioral giveaway. We start with these users as seed bots, and then gradually adapt to identify bots with different behavior patterns.

We explore from these seed bots using the four outlier detection methods described in Section V: LOF and partitioned LOF as our baselines, the ensemble of four anomaly detection algorithms (density-, distance-, angle-, and isolation-based), and this ensemble partitioned across the feature space. We run one round of exploration per seed bot, yielding a total of 15 rounds explored per method, which we call *Level 1 exploration*. We then randomly select 15 followers from our most accurate method with the highest average pairwise percent agreement, *partitioned ensemble*, and perform one round of exploration for each of these 15 seed bots (again using the partitioned ensemble method), which we call *Level 2 exploration*. We repeat this process one more time, and call this *Level 3 Exploration*. The purpose of these multi-level experiments is twofold: to examine how the identity of the seed bot affects precision and to understand how the behavior of the identified bots changes as we explore.

C. Validation

We use four measures to evaluate our results:

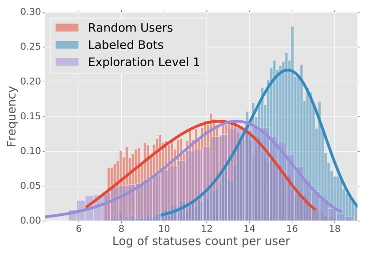
1) User Study: In fraud identification research, one challenge is that the ground truth is, in reality, unknowable. Only the user or botmaster herself truly knows if a given account is a bot. Thus user studies are typically performed to manually validate the results of the algorithm. We base our experimental design on methods from the literature [9][29].

We first conduct a manual examination of a random sample of 20 anomalous users from each of the six experiments to get precision values. The procedure for this examination is as follows: we first train our three annotators by showing them 100 different labeled bot accounts of different types. Next, we have them label each account with 'bot', 'human', or 'unknown'. We take the majority vote for each account as its label and calculate the average pairwise percent agreement, which is where the agreements of all possible pairs are calculated and averaged. Table III shows the results, along with the number of anomalous users identified by BotWalk in each experiment. We estimate that starting from the initial 15 seed bots, over the three rounds of exploration we identify 7,995 new bots, which is over 500 times more than the initial seed bot set size. This estimate is calculated by multiplying the number of BotWalk-identified bots found by the calculated precision for each round and summing them. By scaling these values we avoid over-estimating this number.

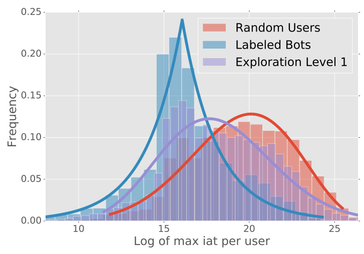
Experiment Type	Number of BotWalk-identified bots	Precision	Average Pairwise Agreement
LOF	3984	60%	67%
Ensemble	3928	65%	80%
Partitioned LOF	3633	90%	87%
Level 1 Exploration (Partitioned Ensemble)	4040	90%	90%
Level 2 Exploration	2215	85%	83%
Level 3 Exploration	3302	75%	87%

TABLE III: Results from the user study for the four anomaly detection methods and the three levels of exploration.

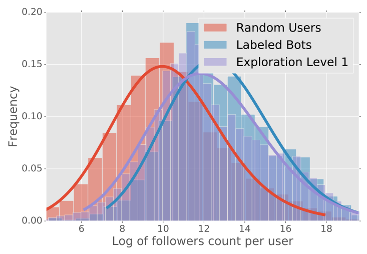
Column 3 in Table III shows that even humans do not always agree on what a bot looks like. However, apart from unpartitioned LOF, which also has low precision, indicating it is not an effective method in this context, all of our experiments have average pairwise percent agreement values of 80% or higher, which is described as an acceptable level of agreement in the literature [30]. Without partitioning, using an ensemble of anomaly detection algorithms improves the precision by 5%. However, when partitioning is included, both LOF and the ensemble have 90% precision. This means that the human annotators agree with 90% of the bot classifications given by our partitioned anomaly detection algorithms. While partitioned LOF and the partitioned ensemble had the same precision scores in this experiment, the ensemble had higher average pairwise percent agreement than partitioned LOF. Furthermore, understanding that bot



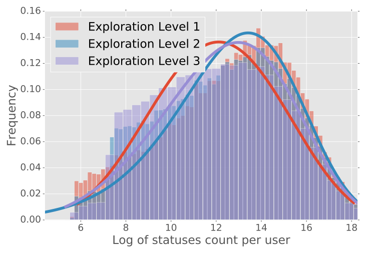
(a) Distribution of log of number of tweets. Random users are modeled by exponentiated norm, prev.-labeled bots by lognorm, and Level 1 by the Beta distribution.



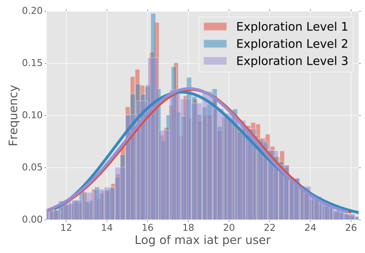
(b) Distribution of log of max inter-arrival times. Random users are modeled by Beta, prev.-labeled bots by log-Laplace, and Level 1 by Beta prime.



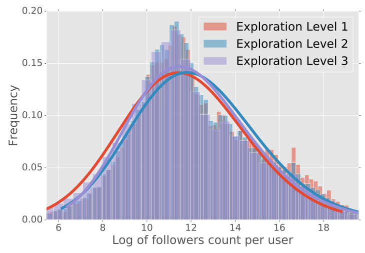
(c) Distribution of log of out-degree of the ego network. Random users are modeled by exponentiated power, while both prev.-labeled bots and Level 1 are modeled by exponentiated Weibull.



(d) Distribution of log of number of tweets. Level 2 is modeled by power-lognorm and Level 3 by exponentiated Weibull.



(e) Distribution of the log of max inter-arrival times. Level 2 is modeled by power-lognorm and Level 3 by Beta prime.



(f) Distribution of log of out-degree of the ego network. Levels 1 and 2 are modeled by exponentiated Weibull.

Fig. 4: Distributions of a small subset of features for random users, Debot-labeled bots, and bots identified by Exploration Levels 1, 2, and 3. Note that our identified bots display different behavior than both random users and previously-labeled bots, and that we are identifying changing behavior as we explore.

behavior continually changes, having an ensemble of techniques may still provide better anomaly detection for future evolving behaviors.

When examining the results for exploration, we can see that as we explore further, the precision stays fairly high. Level 3 Exploration contains the followers of followers of followers of our labeled bots, so the fact that we are still able to identify a high percentage of bots shows that our exploration method is effective. The slight dip in precision suggests that adding some guidance into the system when selecting the seed bots could be helpful. Perhaps choosing a small number of the *most* anomalous followers and only adding those into the seed bank, rather than a random set selected from all fairly anomalous users, could improve performance. We plan to investigate this enhancement in future work.

2) Extreme feature values: To profile the anomalous users identified by BotWalk, we identify how many features each potential bot has that are in the 10th or 90th percentile when compared to a feature vector of 17,000 random users. We limit this evaluation to integer- and floating point-valued features. We then plot the distribution of these values for all of our experiments in Figure 3. Figure 3a includes the percentiles for these random users for reference, and we can see that our identified anomalies have high numbers of 'extreme' features, which shows that our algorithm is successfully identifying anomalous users. Since we know that the features in our comprehensive feature set are effective at identifying bots, these results validate that we are indeed identifying bots in our exploration.

Figure 3b is a zoomed-in version comparing these distributions for the different anomaly detection methods. We can see that the partitioned versions of the outlier detection algorithms are more effective at identifying highly anomalous users than those applied to the feature space as a whole.

To examine how multiple iterations of the exploration process affect the type of anomalous users we are identifying, we compare the extreme feature values for the anomalies found from Level 1, Level 2, and Level 3 Explorations. Figure 3c shows the results of this analysis. We can see that the anomalous users' behavior changes after the first round of exploration, and then remains constant. This makes sense, since for our first round we use Debot-labeled bots as our seed bots, which tend to exhibit very obvious 'dumb' bot behavior, whereas for the following rounds we use randomly-selected anomalous followers as our seed bots.

3) Examination of a subset of feature values: We next wanted to explore the differences in behavior between random users, Debot-labeled bots, and the bots discovered in our three levels of exploration. To perform this comparison, we took a random sample of 10,000 of the previously-labeled bots, 10,000 random users, and the bots identified in each level of exploration and modeled their distributions using the maximum likelihood estimation of the empirical data for the above-described features. We find that BotWalk-discovered bots have significantly different behavior than both Debot-labeled bots and random users for many of our features. Furthermore, as we explore we are finding bots with new behaviors, as evidenced by Figures 4d through 4f.

Experiment	Debot Detection	BotOrNot Detection	Bots Missed
LOF	21%	49%	11%
Ensemble	21%	51%	14%
Partitioned LOF	17%	64%	26%
Level 1 Exploration (Partitioned Ensemble)	19%	60%	30%
Level 2 Exploration	11%	56%	29%
Level 3 Exploration	8.6%	49%	26%

TABLE IV: Detection levels of new bots by known methods

4) Comparison with known methods: We compare BotWalk with the most popular supervised Twitter bot detection algorithm, BotOrNot [14], and the most promising unsupervised algorithm, Debot [13][19], described previously.

Precision: We first evaluate what percentage of our anomalous users each of these methods detect as bots. Table IV shows the results of this study. We see that Debot only identifies between 8.6 and 21% of our anomalous users, which shows that the BotWalk-identified bots found are exhibiting novel behavior that is different from the Debot-labeled seed bots. We calculate the bots missed value based on the BotOrNot classification rate and the precision values from our user study. For example, since 90% of the users identified as anomalous by BotWalk were classified as bots by our human annotators in Level 1 Exploration, and only 60% of those are classified as bots by BotOrNot (using a score threshold of 0.5), we say that this is a miss rate of 30%. Since BotOrNot is a classifier trained on previously-collected data, it makes sense that it is not able to classify bots with novel behavior. These results show that BotWalk is able to identify novel bots at a higher rate when compared with existing methods.

Efficiency: The bot-detection rate of our exploration algorithm necessarily depends on the rate limitations imposed by Twitter. Assuming these are in place, our algorithm is much more efficient at finding bots than any supervised method. Since supervised algorithms need to be given specific accounts, if you select accounts at random and input them to, for example, BotOrNot [14], the bot-detection rate would be, on average, 1469 bots per day. (BotOrNot's query rate is 180 requests per 15 minutes. Choosing a user at random for every query would yield 17,280 users to test per day. Since the current estimate of bots in Twitter is 8.5%, this method would yield on average 1469 bots per day.) Debot needs time to collect and analyze users' data, so it is limited to on average 1619 bots per day [31]. Our method easily beats both of these, identifying on average 6000 bots per day.

VII. Conclusion

This work introduces BotWalk, a near-real time unsupervised Twitter network exploration algorithm that adaptively identifies bots exhibiting novel behavior. We show that anomaly detection is an effective approach to identify bots in an evolving Twitter network and that using domain knowledge to partition the feature space is an effective way to improve precision. We perform experiments to evaluate the performance of an ensemble of outlier detection algorithms, achieving a precision of 90%. We also perform three levels of iterative exploration and show that we are able to identify bots that exhibit different behavior than the seed users at higher precision and efficiency than existing methods.

References

[1] "How Twitter bots fool you into thinking they are real people." http://www.fastcompany.com/3031500/how-twitter-bots-fool-you-into-thinking-they-are-real-people.

[2] P. Galán-García *et al.*, "Supervised machine learning for the detection of troll profiles in twitter social network: Application to a real case of cyberbullying," in *SOCO13-CISIS13-ICEUTE13*, pp. 419–428, 2014.

[3] K. Thomas *et al.*, "Trafficking Fraudulent Accounts : The Role of the Underground Market in Twitter Spam and Abuse Trafficking Fraudulent Accounts," in *USENIX*, pp. 195–210, 2013.

[4] V. S. Subrahmanian *et al.*, "The DARPA twitter bot challenge," *CoRR*, vol. abs/1601.05140, 2016.

[5] J. L. Marble *et al.*, "The human factor in cybersecurity: Robust & intelligent defense," in *Cyber Warfare: Building the Scientific Foundation*, pp. 173–206, 2015.

[6] "Supporting web page containing data, code, case study examples, and user study results." www.cs.unm.edu/~aminnich/botwalk.

[7] R. Ghosh, T. Surachawala, and K. Lerman, "Entropy-based classification of 'retweeting' activity on twitter," *CoRR*, 2011.

[8] J. P. Dickerson *et al.*, "Using sentiment to detect bots on twitter: Are humans more opinionated than bots?," in *ASONAM*, 2014.

[9] "Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?," *IEEE TDSC*, vol. 9, pp. 811–824, Nov. 2012.

[10] C. Teljstedt, M. Rosell, and F. Johansson, "A semi-automatic approach for labeling large amounts of automated and non-automated social media user accounts," *ENIC*, vol. 00, pp. 155–159, 2015.

[11] E. M. Clark *et al.*, "Sifting robotic from organic text: A natural language approach for detecting automation on twitter," *CoRR*, vol. abs/1505.04342, 2015.

[12] J. Pan, Y. Liu, X. Liu, and H. Hu, "Discriminating bot accounts based solely on temporal features of microblog behavior," *Phys. A*, vol. 450, no. C, pp. 193–204, 2016.

[13] N. Chavoshi, H. Hamooni, and A. Mueen, "Debot: Twitter bot detection via warped correlation," in *ICDM*, pp. 817–822, 2016.

[14] C. A. Davis *et al.*, "Botornot: A system to evaluate social bots," WWW '16 Companion, pp. 273–274, 2016.

[15] B. Wang, A. Zubiaga, M. Liakata, and R. Procter, "Making the most of tweet-inherent features for social spam detection on twitter," *CoRR*, vol. abs/1503.07405, 2015.

[16] S. S. A Chakraborty, J Sundi, "Spam: a framework for social profile abuse monitoring," *CSE508 report, Stony Brook Univ.*, 2012.

[17] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *CEAS*, 2010.

[18] M. Giatsoglou *et al.*, "Retweeting activity on twitter: Signs of deception," in *PAKDD*, pp. 122–134, 2015.

[19] N. Chavoshi, H. Hamooni, and A. Mueen, "Identifying correlated bots in twitter," in *SocInfo*, pp. 14–21, 2016.

[20] S. De Paoli, "The automated production of reputation: Musing on bots and the future of reputation in the cyberworld," *IRIE*, 2013.

[21] "Tweepy, Twitter for Python." http://docs.tweepy.org/.

[22] M. V. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *KDD*, 2002.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD*, 2000.

[24] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*, pp. 413–422, 2008.

[25] E. Schubert *et al.*, "On evaluation of outlier rankings and outlier scores," in *SDM*, pp. 1047–1058, 2012.

[26] H. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Interpreting and Unifying Outlier Scores.," in *Sdm*, pp. 13–24, 2011.

[27] https://github.com/amandajean119, title = Amanda Minnich's GitHub.

[28] "Twitter REST API." https://dev.twitter.com/rest/public.

[29] C. M. Zhang and V. Paxson, "Detecting and analyzing automated activity on twitter," in *PAM*, pp. 102–111, 2011.

[30] K. Neuendorf, *The Content Analysis Guidebook*. SAGE Pub., 2002.

[31] N. Chavoshi, H. Hamooni, and A. Mueen, "Temporal patterns in bot activities," in *TempWeb*, 2017.