BOTSHAPE: A NOVEL SOCIAL BOTS DETECTION APPROACH VIA BEHAVIORAL PATTERNS

Jun Wu[1], Xuesong Ye[2] and Chengjie Mou[2]
[1]College of Computing, Georgia Institute of Technology, Atlanta, United States
[2]Department of Engineering and Technology, Trine University, Phoenix, United States

ABSTRACT

An essential topic in online social network security is how to accurately detect bot accounts and relieve their harmful impacts (e.g., misinformation, rumor, and spam) on genuine users. Based on a real-world data set, we construct behavioral sequences from raw event logs. After extracting critical characteristics from behavioral time series, we observe differences between bots and genuine users and similar patterns among bot accounts. We present a novel social bot detection system BOTSHAPE, to automatically catch behavioral sequences and characteristics as features for classifiers to detect bots. We evaluate the detection performance of our system in ground-truth instances, showing an average accuracy of 98.52% and an average f1-score of 96.65% on various types of classifiers. After comparing it with other research, we conclude that BOTSHAPE is a novel approach to profiling an account, which could improve performance for most methods by providing significant behavioral features.

KEYWORDS

Social bots detection, behavioral features mining, machine learning

1. INTRODUCTION

1.1. Background and Motivation

It had become common sense that bot accounts flooded almost every popular social network platform. Many research works have realized these issues and made solutions like Twitter [1], Facebook [2], and Reddit [3]. Along with the breaking out social bot population, user-experience of normal users deteriorate quickly due to malicious bot direct or indirect disturb, e.g., fake news [4], spam [5], rumor [6], and misinformation [7]. Furthermore, it will finally push platforms to improve a large amount of maintenance and risk control costs or otherwise suffer from user loss. Therefore, designing highly accurate and easily deployed bot detection systems is substantial for research and industry.

1.2. Bot Detection Approaches

Recent social bot detection works could be categorized into three strategies: account-based, content-based, and graph-based.

Account-based approach mainly focuses on mining risk signatures as features from account meta-data and statistical indicators for machine learning algorithms to do classification. Kai-

David C. Wyld et al. (Eds): ICAITA, CDKP, CBW, SOFT, CMC, Signal, MLT-2023
pp. 45-60, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130604

Cheng *et al.* [8] extracted two types of features, including raw data like follower count and derived features like follower growth rate and the length of the screen name. Saleh *et al.* [9] summarized a bunch of features according to account information like age, length of the name, and count of followers and then used a support vector machine to separate bots from genuine users. Hrushikesh *et al.* [10] ensemble the prediction results of three machine learning models to improve the classifying accuracy after a comprehensive account features collection, e.g., location, profile image, and daily average tweet count, and so on. Maryam *et al.* [11] utilized GloVe [12] to create account embedding based on age, gender, education, and personality from meta-data, which firstly proposes the user embedding concept.

Content-based method systems shift the spotlight to public texts posted by users, like tweets, and apply natural language processing techniques to mine risk words or embedding semantics. The primary assumption of those works is that content posted by bots tends to uncover their fraud intentions, and various intentions are clusters in high-dimensional language embedding space. Anisha *et al.* [5] applies the TF-IDF and Bag of Words technique to generate text features for the downstream machine learning model to train and predict. BGSRD [13] combined Bert [14] and GCN(Graph Convolutional Networks) [15] algorithm to jointly learn representation from multiple historical tweets of each account for bot classification tasks. DeepSBD [16] learned text representation based on historical tweets and mixed content embedding with account features via joint representing. Maryam *et al.* [4] also uses BERT to generate text embedding from tweets, showing a significant performance in detecting fake news about the COVID-19 topic.

Graph-based approach gets popular in the bot detection domain after the rapid evolution of the graph representing techniques. After 2016, newly proposed algorithms GraphSAGE [17] and GCN [15] perform significantly among various network relationships in the real world, like social networks, online shopping, and citation map. The graph-based approach reuses useful account-based and content-based features by transforming them into node embedding. Moreover, it explores the optimal network topology to share and transfer node information among neighbors, outperforming traditional methods. Some recent works show a start-of-art accuracy by designing appropriate graph structures and node features. Seyed *et al.* [18] firstly apply graph convolutional neural networks to learn one node's representation based on account features of itself and its neighbors. Shangbin *et al.* [19] applied GCN algorithm on the user following relationship graph, then represented raw node features including user profile, categorical and numerical data of account activity. Shangbin *et al.* [20] constructs two kinds of heterogeneity structures, including relation and influence, leveraging the topology to identify the difference between genuine users and social bots.

1.3. Existing Problems

Most *account-based works* focus on evaluating the effects of different machine learning models without a deep digging into the behavior patterns of bots. **The first problem** is that most works tend to input coarse-grain statistical indicators as features into models without controlling variables. For example, the count of followers is an essential feature in many bot detection models. However, the count of followers of one account will naturally increase and accumulate day-to-day after registering. Engineers usually calculate the indicators on the day of building models. However, the registering dates of accounts are different. As a result, these calibrated features become noisy data to prevent the classifier from making an accurate prediction. In short, adequate log data is not utilized sufficiently in this way. **The second problem** is that most works consider bots isolated rather than gangs of attackers. Consequently, little research is focusing on discovering behavior similarity among bot accounts. In graph-based methods, accounts will exchange node information with neighbors. However, most focus on finding a better network topology rather than exploring behavioral node features.

1.4. Our Contributions

To summarize, this paper makes the following contributions:

– We first propose a novel type of feature to profile a social account's behavioral pattern, supported by a solid measurement work to observe distribution divergence between genuine users and bot accounts. Based on behavioral sequences, we adopt proper algorithms to catch their significant patterns for prediction.

– We design a bot detection system BOTSHAPE integrating an automatic behavioral feature generation process and implementing a complete pipeline log processing, feature engineering, andprediction. Input parameters are simplified into a minimal range to improve the automation degree of the whole process.

– BOTSHAPE performs high accuracy, with an average accuracy of 98.52% and an average f1-score of 96.65% in evaluating experiments across four classifiers and three ground truths. It presents a steady performance improvement compared to *account-based features* and also shows the crucial role of extractingbehavioral patterns. BOTSHAPE is easy to combine with other bot detection systems by providing compelling behavioral features. For example, *graph-based approach* could use behavioral features as the node features and spread pattern information into the whole network..

2. DATASET

Table 1. Key attributes of account and tweeting data in CRESCI2017 data set.

Attribute	**Descriptions**
User ID	It refers to the unique identifier for each tweet account.
Created At	It referred to the time stamp at the time of the account registration.
Account Information	There are several essential attributes, such as the nickname of an account, the personal introduction, whether the avatar is the default, and the common location.
Interaction (user-level)	They refer to the number of accounts that interact with an account, potentially reflecting the social vitality of an account, such as the count of followers, friends, and favorites.
Tweet ID	It refers to the unique identifier for each tweet post. It has a unique id of its author.
Tweet Created At	It is the time stamp of the posting time of a work like a tweet.
Post Content	It refers to the raw text or voice of a post.
Interaction (tweet-level)	They are statistics of re-tweeting and replying to a post, reflecting its popularity.

Cresci *et al.* [1] published a data set of Twitter (called CRESCI2017) of four types of accounts, including genuine users, social bots, traditional users, and fake followers. It collected user information and event logs (mainly tweeting records with timestamps, identifiers, and text) in the real world. The total number of social bots is 4912, consisting of three small data sets collected at different periods (from the easiest tweet publish time to the last one): (i) from March 17, 2009, to May 26, 2014 (ii) from September 9, 2008, to March 22, 2014 (iii) from September 14, 2008, to April 11, 2014. The data group of genuine users begins on January 22, 2007, and ends on April 20, 2015, with 3474 accounts in total (only 1083 of them have tweeting logs). There are two extra spam account data sets with complete tweeting logs: (i) traditional spam bots: from July 4, 2007, to March 8, 2010, with 1000 accounts that have tweeting logs (ii) fake followers: from December 7, 2007, to April 30, 2013, with 3351 accounts. The collected tweets come across a very long

period, sufficient to observe accounts' early and long-term behavior. CRESCI2017 supports abundant real-world user and tweet data for analysis and modeling. Based on domain knowledge, we highlight important ones and summarize them in Table 1.

3. MEASUREMENT

In our work, we commit to solving the two problems discussed in sub-section 1.3, through refined behavior indicators and solid measurement results. We discover characteristics of accounts' behavior patterns and uncover hidden correlations between and show how different they are between bot accounts and genuine users. All measurement results support a proper architecture of BOTSHAPE in Section 4.1, including feature extraction and prediction model.

We split measurement work into two parts: (i) The first part shows how to construct calibrated indicators without a life-cycle inconsistency problem. Then we analyze how the calibrated indicators change over time on a real-world data set. Furthermore, we also compare the fluctuating pattern between bot accounts and genuine users. (ii) The second part digs into tweet count analysis from a time series perspective. We observe the busy and idle time in the one-day and one-week range. Then we observe the clusters of behavioral time series to verify the assumption that bots have more synchronized behaviors than genuine users because of the centralized control of the dark industry.

3.1. Account Life-Cycle and Data Cleaning

Account life-cycle and event logs: Each social network account has its life-cycle, starting from the registration time to the current system time. One account did many actions in its life-cycle, e.g., following, liking, posting, restored as event logs by the system. Behavioral logs accumulate day-by-day as time goes on; as a result, indicators like the count of tweets increase and change. Registration time-stamp is a vital starting point to calibrate event logs. We construct a new metric $\boldsymbol{Indicator_{dur}}$ using $\boldsymbol{dur}$ as a variable that refers to the duration from the registration to the current date-time. For fairness, we only compare indicators among accounts with the same parameter dur.

With complete tweeting records, we could calculate multiple behavior indicators with different dur parameters at various positions in one account's life-cycle. For a better observation, we select multiple dur variables with the same time interval. The definition of dur in the measurement is the count of days from the registration to the analysis date-time. For example, if one account registered on May 1, 2014, and the virtual analysis happened on May 3, 2014, the dur equals two days.

Removing inactive accounts: We calculate Indicatordur for each account after removing the inactive ones. We define active account as one having at least one tweet in the first month (30 days) after registering. Inactive accounts, most are audiences, present less behavior. They occupy a large proportion of the data set. We remove them to focus on informative and active behavior data rather than letting inactive accounts average and weaken important distribution regularity. For a fair comparison, we remove inactive accounts from the bot accounts and normal users simultaneously.

3.2. Observe Behavioral Indicators Distribution in the Life-cycles of Accounts

Behavioral statistic in a time-window. The data set CRESCI2017 has complete tweeting records with time stamps, contexts, identifiers of accounts, and registering time stamps.

Therefore, we set the analyzed indicator as the tweets count for each account in this section. We consider measuring tweeting behavior enough because the account independently controls tweeting behavior having no relationship with other accounts, unlike the count of followers and friends affected by other accounts. Therefore, it is the most direct and meaningful attribute to show accounts' behavior and capture attack traces.

To observe how one behavioral indicator fluctuates, we select various window sizes to calculate it. We first select one month as the time interval, meaning dur equals 30, 60, 120, ... 360 (days). We also select one day as the time interval for a more fine-grain analysis, where dur equals 1, 2, 3, ... 30 (days). Indicatordur equals the tweets count from the registration date to the dur day. We analyze how an indicator time series(called Seqbhv) fluctuates across 12 months and 30 days. For a specific dur, we use box-plot [21] to present the distributions of indicatordur consisting of all accounts. In the box-plot figure, each box shows the minimum value, first quartile, average, third quartile, and maximum value, which could clearly show the range and critical statistics of a bunch of data points.

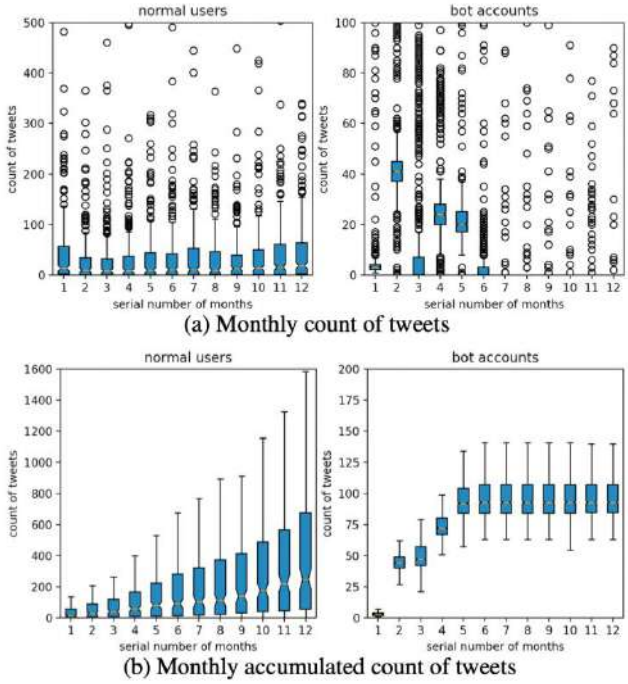

Figure 1. Boxplots of tweeting counts: bot accounts vs normal users

A long-term observation. Figure 1 shows the box-plot of monthly $Indicator_{dur}$. The upper left figure uses the data of normal users, and the upper right figure plots the data of social bot accounts. When comparing the two figures, we conclude that normal users are continually active, but bot accounts cooperate tacitly to reduce and even stop posting tweets after the sixth month. The following two figures are the accumulated tweet count from the registration to the corresponding month. The box plots of normal users show steady growth, but plots of bot accounts show a sudden termination starting in the fifth month.

All the results reflect each role of accounts. Most regular users have one account for one platform and persistently use it. In an online social occasion, changing into a new account tends to mean losing followers and friends, so regular users nearly do not change accounts. However, along with doing more and more spam, bot accounts will get more easily caught by the platform's risk

control systems. What follows is that the cost of escaping detection and unlocking account improve quickly. As a result, the controller chooses to give up old bot accounts and register product new bot accounts for new spam purposes in the future.

The period at newly-registration. The monthly *dur* interval is a bit coarse, in which account behavior patterns in newly-registered periods would hide. Therefore, we construct daily *dur* indicators to watch the situation in the first month after registration. Figure 2 is the box plot of tweeting count distribution at week-grain for three months (21 weeks). The two upper figures are the weekly count of tweets, and the two below are the accumulated count. The differences between bots and normal users are similar to monthly time granularity.

The normal users' tweeting box plot shows a burst of tweets in the first week, which means normal users often post more tweets at registration. However, bot accounts unexpectedly stay freezing (no tweeting behavior) until the third week after registration. Now look at the accumulation box plot: normal users show an apparent steady increase in tweet counts, but bot accounts have an unusual stopping of tweeting from the eighth week to the thirteenth week. In addition, the tweeting count of one normal user is larger than bot accounts on average.

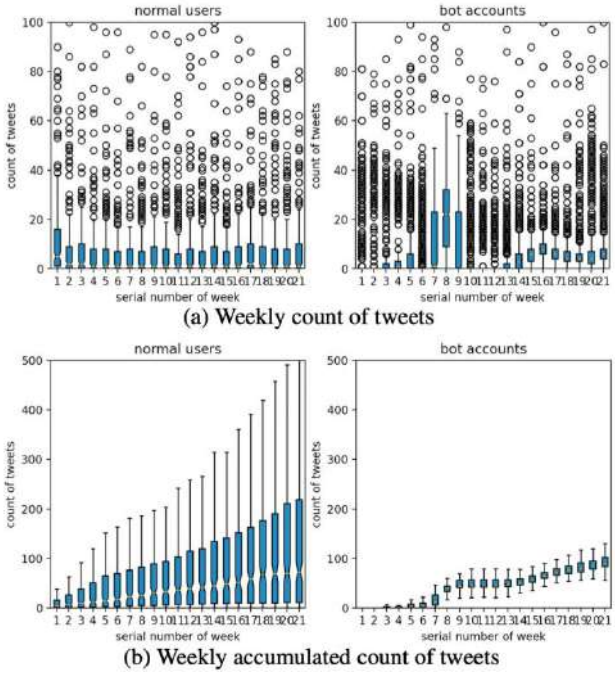

Figure 2. Boxplots of tweeting counts: social bots vs normal users

3.3. Mine Effective Patterns from Behavioral Sequences

Busy and idle in a time-series seasonality. Due to the regularity of using habits, user behavior statistics in social network platforms are often time series with seasonality. Also, the busy and idle time distribution are related to usage habits. For example, people sleep at night and then perform an idle period. We analyze the tweeting behavior time series properties based on a seasonal method. We group tweets into multiple parts according to time attributes, such as the hour of the day (from 0 am to 11 pm) and the day of the week (from Monday to Sunday).

Figure 3 shows the results. We use the proportions of the counts occupying the total amount rather than the actual tweeting counts. The peak and low of normal users and bots are different. Normal users peak at 5 am and low around 11 am. Bot accounts peak at 5 pm and low at 6 am.

What is more different is that the trends of the two groups are nearly opposite, from 0 am to 9 am, which means normal users are active, but bots are inactive. The day-of-week distribution reflects a similar problem. We observe the conflict between peak and low and different local trends. Tweeting user count distributions in 24 hours and seven days are uniform and similar, whether normal users or bot accounts. It means the active proportions of accounts in the two groups are similar in each time unit.

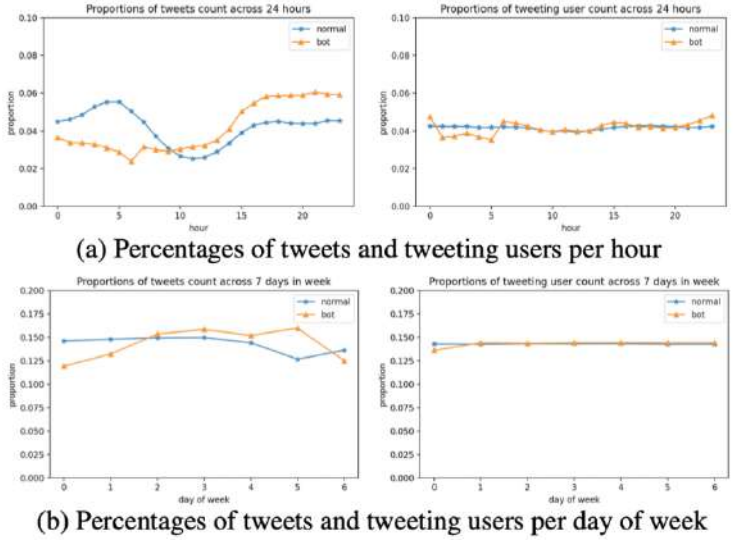

Figure 3. Decomposed seasonality from behavioral sequences of tweeting

Patterns of time series. The wave's shape is critical for classifying and clustering time series. Several algorithms could extract shaping features, such as Dynamic Time Warping (DTW) [22], Shapelets [23] and deep learning method. [24]. We choose the DTW algorithm to cluster monthly tweeting count time series because it is highly efficient to discover similar time series patterns among multiple time series. We use tslearn [25] k-means and dynamic time warping library for computing. It uses DTW as core features in k-means algorithms and can group a large amount of time series into clusters quickly and precisely.

For fairness, we randomly sample 200 tweeting behavior time series from the normal user and bot account groups, respectively. Figure 4 and 5 show the result. It is obvious that the time series of bots are highly similar, reflected in that in each cluster, shapes of curves are similar, and overalloutlines (thick black lines in each cluster) of 5 clusters are similar. The clustering results of normal users are the opposite, with a phenomenon where in each cluster, curves are mussy, having no similarity without a clear overall outline. Also, there is nearly no similarity or correlation among the 6 clusters. We can conclude that bot accounts' behavior correlates and acts concurrently due to centralized control. Regular accounts, controlled by natural persons, perform vastly different behavior patterns due to various usage habits.

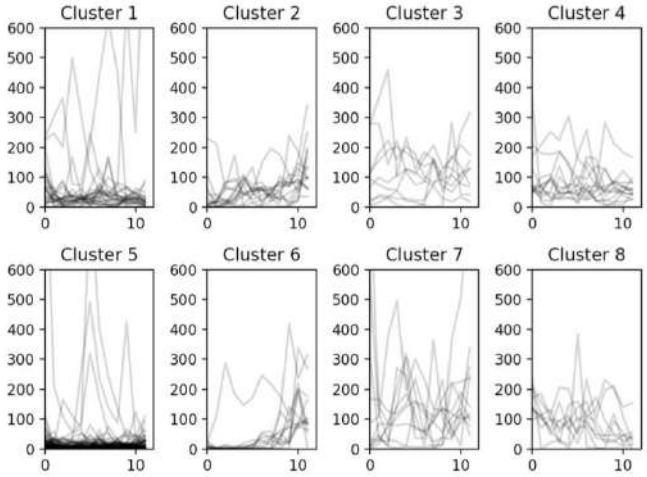

Figure 4. Clusters of monthly behavioral sequence for normal users

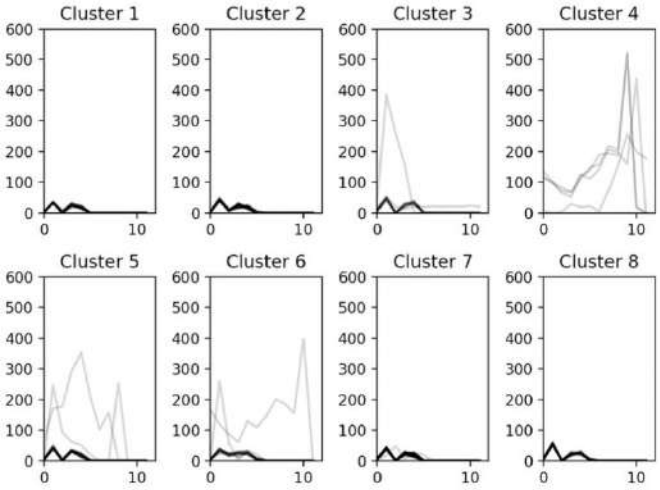

Figure 5. Clusters of monthly behavioral sequence for social bots

All the indicators shown in measurement results could be transformed into account features for machine learning models to do classification. Also, the variances of those features between a bot and normal users could improve the separability of the two data points groups, making prediction more accurate.

4. DESIGN

We present BOTSHAPE, a social bots detection framework. It takes account logs as input, does an automatic behavioral feature-generating process, and then detects social bots based on machine learning classifiers. The feature engineering process has two steps: (i) extract behavioral sequences from account registration logs and event logs in various time intervals. (ii) compress raw behavior sequences time series to seasonality attributes and distinctive shapelets features [23]. Behavioral features are actual multiple numerical vectors compatible with many machine learning classifiers.

4.1. Main Idea

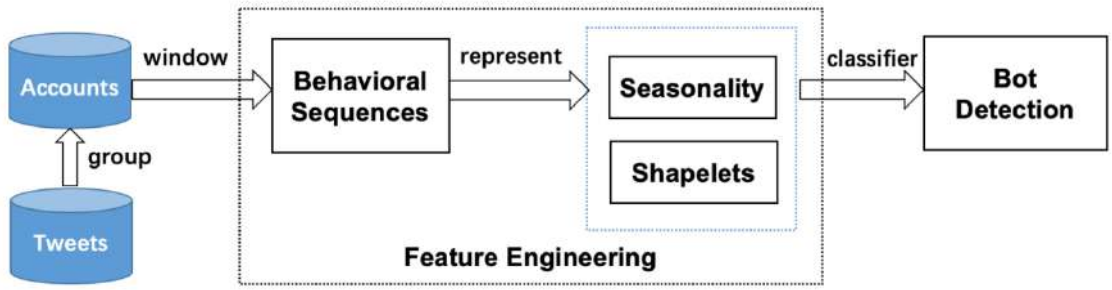

Figure 6. BOTSHAPE architecture

How to detect bots at an earlier time? Figure 6 shows the BOTSHAPE architecture, where each module is reasonable and has a corresponding supporting point in section 3. BOTSHAPE takes full advantage of the raw account event logs to discover the most compelling features for a classifier to distinguish bot accounts and user accounts. Different from **account-based** method, taking indicators accumulated for a more extended period as input, BOTSHAPE could be deployed anytime after the registration due to its fine-grain observation view.

How to compress dimensionality and keep useful features? Extracting the best feature set is not trivial because the original features constructed by setting various time-related parameters are high-dimensional, leading to the Curse of Dimensionality problem [26]. In other words, it will not give rise to a high detection accuracy if only piling up a mass of features. Therefore, it is necessary to mine prominent features from many raw behavioral series for the downriver model to make prediction.

How to design an automatic system? We assess the automation degree of a system based on the complexity of the manual parameter-setting process. Namely, it is highly automotive if a system could directly produce compelling features after setting a few parameters without too many attempts. We design BOTSHAPE according to this principle. BOTSHAPE applies a two-step separate feature engineering process with flexible parameter management, decoupling features election from raw feature production. The first step is generating a batch of statistical behavior sequences via different parameter settings such as time interval, period, and statistic functions. The second step focuses on mining the critical fluctuation points of those sequences based on the Shapelets method, which vastly reduces the dimensionality of feature vectors. Users could tune parameters respectively and flexibly for the two parts.

4.2. Behavior Sequence Features

We now elucidate the first step of the feature engineering process about generating behavior sequences using event logs $\boldsymbol{Log_{bhv}}$and registration information$\boldsymbol{Log_{reg}}$. We format behavior sequences as multiple time series data. The program extracts behavioral time series from raw logs based on two time-related parameters. Parameter $\boldsymbol{dur}$ is the duration from the registration to the computation time. Parameter $\boldsymbol{gran}$ refers to the time granularity, such as day, week, and month, which is the window size for calculating statistical features like the number of tweets.

Algorithm 1 Generate Behavioral Sequences (Time Series)

Require:
1: RegistrationLogs: $Log_{reg} = (id_{user}, t_{reg})$
2: EventLogs: $Log_{bhv} = \{(id, t_1), (id, t_2), \ldots, (id, t_n)\}$
3: Parameters: dur, gran
Ensure:*Behavioral Time-series:*ts
4: function $gen\_bhv\_sequence(Log_{bhv}, t_{reg}, dur, gran)$
5: win = floor($\frac{dur}{gran}$)
6:ts = **new** int[win]
7: **for** k = 1, ...win**do**
8: $T_{st} = gran * (k - 1)$
9: $T_{ed} = gran * k$
10: $cnt = 0$
11:**for** i = 1,2,...,n **do**
12:**if**$t_i - t_{reg} > T_{st}$**and**$t_i - t_{reg} <= T_{ed}$**then**$cnt = cnt + 1$
13:**end if**
14:**end for**
15: $ts[k] = cnt$
16:**end for**
17:**return** ts
18: **end function**

The pseudo-code introduces how to generate behavioral sequences in a time series format. It takes two kinds of data as input, including the registration time stamp of an account and its event logs. Event logs could be any action on the social platform, like FOLLOW, SHARE, POST, and EDIT PROFILE. The computing is individual for each account, so this process can be deployed paralleled and distributed. In the generation process, it firstly computes the count of windows $\boldsymbol{win}$, equaling $\boldsymbol{dur}$ dividing$\boldsymbol{gran}$, then it scans each time window to calculate the behavioral statistic like the count of tweets. In each round, the program sets the start time $\boldsymbol{T_{st}}$ and end time of $\boldsymbol{T_{ed}}$each window for precisely distributing each $\boldsymbol{Log_{bhv}}$item to its time window. BOTSHAPE outputs the sequences of behavioral statistics in chronological order, forming a time series. Engineers could generate a bunch of time series by setting various function parameters.

4.3. Behavior Pattern Features

Seasonality Decomposition. In a time series data, seasonality is the periodic fluctuation correlated strongly to the time attribute. Some time series data could perform a seasonal variation because the related entity changes regularly over time. For example, the number of online users in a network correlates with the working and sleeping time (busy or idle state) of people [27], so the time series indicator could perform a repeat and similar pattern. Plus, social platforms for online activity are highly affected by user usage habits and tend to present hourly, daily, and weekly seasonality. Also, in sub section 3.3, we compare the behavioral seasonality distributions of bots and genuine users, hourly and daily, respectively, showing an evident divergence.

Assuming that classifiers could separate bots from genuine users after exploring the seasonality of behavioral sequences, BOTSHAPE owns a continued process of seasonality extraction. In detail, After generating a time series, BOTSHAPE continues to compute the seasonality, which is still a sequence where each item is the mean value of corresponding elements. For example, for extracting the weekly seasonality extraction from a day-gran time series, BOTSHAPE firstly queries the day of the week of each time-stamp and then allocates them to seven groups (Sunday, Monday, ..., Saturday), finally averages each group and organizes them into a new sequence in order.

Shapelets Representation Shapelets [23] are subsequences of a time series that are prominently distinct characteristics of its class. The Shapelets algorithm performs outstandingly in time series classification problems compared to raw data. The algorithm targets finding the best splitting strategy for maximizing the information gain (difference between entropy before and after the splitting).

In sub section 3.3, we observe an apparent similar time series shape in six clusters of bot behavioral sequences. If BOTSHAPE could represent the pattern correctly, it would vastly improve the detection accuracy for bots. Therefore, BOTSHAPE also takes continued Shapelet mining after constructing behavioral sequences. It applies a python package called tslearn [25] to learn shapelets. It learns from the raw time series and their labels from the train set to discover the best splitting points for the most accurate classification and then extracts the shapelets of the time series in the test set via the same splitting manner. In the default setting, BOTSHAPE extracts shapelets from weekly and monthly behavioral sequences as new features, with a default subsequence length setting of 30% of sequence length for weekly and 50% for monthly.

4.4. Bot Detection

Behavioral data constructed by BOTSHAPE, including behavioral sequence, seasonality, and shapelets, are potential features for machine learning classifiers to fit and predict. We emphasize that BOTSHAPE focuses on behavioral feature engineering, providing extra general bot features, rather than **account-based** attributes. To evaluate its general utility, BOTSHAPE integrates multiple prediction algorithms instead of fixing the classifier. We evaluate prediction accuracy in sub-section 5.2 shows behavioral features all perform a very high detection accuracy when using different classifiers to predict. Eventually, secondary time series features (seasonality plus shapelets) perform better.

5. EVALUATION

5.1. Ground-Truth and Metrics

Ground-truth is information about known properties of some entities based on observation and expert knowledge. In the classification problem, it provides the precise class of each instance (called **label**). Ground truth plays a vital role in machine learning because it is a benchmark for evaluating performance and finding the best from models with various algorithms, parameters, and features. The data set CRESCI2017 has three types of bot accounts, so we reconstruct them into three ground-truth sets shown in Table 2. Each set consists of bot accounts and genuine users, and we define bots as positive instances and genuine ones as negative instances. One set by one, we gradually escalate the scope of bots for a more detailed performance evaluation in three situations.

Table 2. Ground-truths.

Set	Positive	Negative	Description
BotSet1	4912	1083	social bots, genuine users
BotSet2	5912	1083	social bots, traditional bots, genuine users
BotSet3	9263	1083	social bots, traditional bots, fake followers, genuine users

In a standard machine learning classification pipeline, engineers randomly split the instances of a ground truth into two parts, including train set occupying 70 percent and test set occupying 30 percent. BOTSHAPE uses instances of the train set to fit the functional relationship between features and actual labels. Then it outputs predicted labels for instances in the test set after in-

putting features. Due to the difference between actual labels and predicted labels in the test set,there are four cases for each instance: (i) True Positive (TP): the actual and the predicted label are all positive; (ii) False Positive(FP): the actual label is negative, and the predicted is positive; (iii) True Negative (TN): the actual and the predicted label are all negative; (iv) False Negative (FN): the actual label is positive, and the predicted is negative.

We adopt two metrics **accuracy** and **f1-score** based on those four statistics. **Accuracy** equals $\frac{TP+TN}{TP+FP+TN+FN}$. It reflects the correct rate of all predictions, regardless of whether the class is positive. Generally, the higher the accuracy, the more right prediction occurs in an onlinesystem. **F1-score** combines two important metric precisionandrecall, equaling $\frac{2 \cdot precision \cdot recall}{precision+reall}$. Precisionis the rate of correct prediction in predicted positive instances, equaling $\frac{TP}{TP+FP}$. Highprecision refers to a lower false alarm rate, meaning fewer wrongly punished genuine users. Recallis the bot discovery rate in actual positive instances, and it equals $\frac{TP}{TP+FN}$. Higherrecall stands fora less count of bots that the detection system will miss. F1-score could comprehensively reflects precision andrecall performance, by averaging them. In general, the higher f1-score is, the higher precision and recall are.

5.2. Assess Effectiveness of Behavioral Features

We evaluate the bot detection performance of BOTSHAPE three ground-truth. We split feature data into three groups: **account**, **sequence**, and **pattern**. **Account features** refer to user properties profiling an account, including five indicators directly coming from data set CRESCI2017: the count of statuses, the count of followers, the count of friends, the count of favorites, and the count of lists. Also, for fair competition, we add one indicator into the feature set: the total count of tweets within one year after registration. **Sequence features** (mentioned in sub section 4.2) consists of 3 types of raw behavioral sequences, including daily tweet count in the first 30 days, weekly tweet count in the first 53 weeks, and monthly tweet count in the first 12 months after registration. **Pattern features** (mentioned in sub section 4.3) have two significant features compressed from behavior sequence feature set, including *seasonality* and *shapelets* features.

Table 3. Accuracies and F1-scores on BotSet1.

data set		**social bots**			
feature set		**account**	**sequence**	**pattern**	**gain**
SVM	**Accuracy**	86.56%	96.82%	**97.83%**	**11.27%**
	F1 Score	79.65%	94.45%	**96.11%**	**16.46%**
LR	**Accuracy**	94.95%	86.46%	**98.84%**	**3.89%**
	F1 Score	90.68%	64.90%	**97.92%**	**7.24%**
MLP	**Accuracy**	82.97%	84.34%	**98.94%**	**15.97%**
	F1 Score	45.35%	54.01%	**98.11%**	**52.76%**
RF	**Accuracy**	97.22%	86.81%	**98.69%**	**1.47%**
	F1 Score	94.97%	68.55%	**97.68%**	**2.71%**

Table 4. Accuracies and F1-scores on BotSet2.

data set		social + traditional bots			
feature set		account	sequence	pattern	gain
SVM	Accuracy	80.94%	93.16%	**95.71%**	**14.77%**
	F1 Score	74.44%	88.55%	**91.90%**	**17.46%**
LR	Accuracy	94.41%	84.71%	**97.27%**	**2.86%**
	F1 Score	88.13%	57.30%	**94.90%**	**6.77%**
MLP	Accuracy	84.06%	84.06%	**96.80%**	**12.73%**
	F1 Score	45.67%	45.67%	**93.98%**	**48.31%**
RF	Accuracy	96.84%	97.62%	**98.96%**	**2.12%**
	F1 Score	93.68%	95.44%	**98.01%**	**4.32%**

Table 5. Accuracies and F1-scores on BotSet3.

data set		all bots + fake followers			
feature set		account	sequence	pattern	gain
SVM	Accuracy	90.70%	96.32%	**97.18%**	**6.48%**
	F1 Score	80.13%	90.63%	**92.28%**	**12.15%**
LR	Accuracy	95.75%	89.54%	**98.57%**	**2.82%**
	F1 Score	86.40%	58.77%	**96.23%**	**9.83%**
MLP	Accuracy	89.54%	89.54%	**98.93%**	**9.39%**
	F1 Score	47.24%	47.24%	**97.17%**	**49.93%**
RF	Accuracy	98.19%	98.51%	**99.17%**	**0.98%**
	F1 Score	94.96%	95.98%	**97.73%**	**2.76%**

To prove that features constructed by BOTSHAPE feature engineering approach (**sequence features** and **pattern features**) have steady improvements on different classifiers, we select four widely used algorithms: Support Vector Machine [28] (SVM), Logistic Regression [29] (LR), Multilayer Perceptron [30] (MLP) and Random Forest [31] (RF). We apply a famous machine learning python library scikit-learn [32] to implement all the classifiers.

Table 3, 4 and 5 show the accuracy and f1-scores on all classifiers and feature groups across three groundtruths. The result points out **pattern features** perform best on two metrics across all classifiers, with all accuracy exceeding 97% and a very high f1-score range from 92% to 98%. **Sequence features** also present high accuracy on SVM and RF, but it shows low f1-scores on classifier Logistic Regression and Multilayer Perceptron. We infer that irrelevant indicators in **behavior features** become noise, making a simple classifier learn fake functional relationships between noisy features and labels. Random Forest and Support Vector Machine are more robust to noisy features.

5.3. Compare with Other Approaches

As sub-section 1.2 mentioned, there are three detection approaches: **accout-based**, **content-based**, and **graph-based**. Under the condition of CRESCI2017 data set, we could contrast BOTSHAPE with **account-based features**. We could not experiment with the **Graph-based** method because there are no edges like the following relationship in the data set. We do not implement **content-based** method because our approach has already achieved very high accuracy, and its used computation resource is very saving compared with content models.

In table 3, 4, and 5, we define a new metric called ***performance gain*** to measure the improvement of BOTSHAPE, which is the difference between *pattern features* and *account features*. The value of *performance gain* is influenced by the actual accuracy of BOTSHAPE and

the initial accuracy of *account features* because the largest accuracy value is 100%. F1-score is also the same. *Gain of accuracy* ranges from 0.98% to 11.27%, with an average value of 6.53%. *Gain of f1-score* ranges from 2.71% to 52.77% with an average value of 19.23%.

6. CONCLUSIONS

We study the problem of detecting social bots in online social network platforms, inspired by novel account behavioral features mining and characterizing on a real-world dataset collected from TWITTER. We present BOTSHAPE, an intelligent social bots detection framework consisting of three processes: account event logs pre-processing, behavioral sequences and patterns mining, and accurate bot prediction. We target discovering novel but compelling fine-grain behavioral features which have never been systemically studied and evaluated by previous works. BOTSHAPE first automatically generates behavior sequences, which are time series of statistics of different time-window, to profile various periods in one account's life-cycle. Based on apparent similarities of the characteristics and fluctuations of behavioral sequences after analyzing and clustering, our system second compresses raw sequences from high-dimensionality vector to low-dimensionality but more effective time series patterns based on the Shapelet algorithm. BOTSHAPE was evaluated as generally accurate on three ground truths and four classification algorithms. Also, behavioral patterns as features also show an outstanding performance than account-based features and raw behavioral sequences produced by the first process.

REFERENCES

[1] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 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, pages 963–972, 2017.

[2] Giovanni C Santia, Munif Ishad Mujib, and Jake Ryland Williams. Detecting social bots on facebook in an information veracity context. In Proceedings of the international AAAI conference on web and social media, volume 13, pages 463–472, 2019.

[3] Sofia Hurtado, Poushali Ray, and RaduMarculescu.Bot detection in reddit political discussion.In Proceedings of the fourth international workshop on socialsensing, pages 30–35, 2019.

[4] Maryam Heidari, Samira Zad, Parisa Hajibabaee, Masoud Malekzadeh, SeyyedPooya HekmatiAthar, Ozlem Uzuner, and James H Jones. Bert model for fake news detection based on social bot activities in the covid-19 pandemic. In 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pages 0103–0109. IEEE, 2021.

[5] AnishaP Rodrigues, Roshan Fernandes, Adarsh Shetty, Kuruva Lakshmanna,RMahammad Shafi, etal.Real-time twitter spam detection and sentiment analysis using machine learning and deep learning techniques. Computational Intelligence and Neuroscience, 2022, 2022.

[6] Zhen Huang, Zhilong Lv, Xiaoyun Han, Binyang Li, Menglong Lu, and Dongsheng Li. Social bot-aware graph neural network for early rumor detection. In Proceedings of the 29th International Conference on Computational Linguistics, pages 6680–6690, 2022.

[7] McKenzie Himelein-Wachowiak, Salvatore Giorgi, Amanda Devoto, Muhammad Rahman,Lyle Ungar, HAndrew Schwartz, DavidH Epstein, Lorenzo Leggio, and BrendaCurtis. Bots and misinformation spread on social media: Implications forcovid-19. Journal of medical Internet research, 23(5):e26933, 2021.

[8] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. Scalable and generalizable social bot detection through data selection. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 1096– 1103, 2020.

[9] Saleh Beyt Sheikh Ahmad, Mahnaz Rafie, and Seyed Mojtaba Ghorabie. Spam detection on twitter using a support vector machine and users' features by identifying their interactions. Multimedia Tools and Applications, 80(8):11583–11605, 2021.

[10] Hrushikesh Shukla, Nakshatra Jagtap, and Balaji Patil. Enhanced twitter bot detection using ensemble machine learning. In 2021 6th International Conference on Inventive Computation Technologies (ICICT), pages 930–936. IEEE, 2021.
[11] Maryam Heidari, JamesH JonesJr, and Ozlem Uzuner.Online user profiling to detect social bots on twitter.arXiv preprint arXiv:2203.05966, 2022.
[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.
[13] Qinglang Guo, Haiyong Xie, Yangyang Li, Wen Ma, and Chao Zhang. Social bots detection via fusing bert and graph convolutional networks. Symmetry, 14(1):30, 2021.
[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
[16] Mohd Fazil, Amit Kumar Sah, and Muhammad Abulaish. Deepsbd: a deep neural network model with attention mechanism for socialbot detection. IEEE Transactions on Information Forensics and Security, 16:4211–4223, 2021.
[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.
[18] Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. Detect me if you can: Spam bot detection using inductive representation learning. In Companion Proceedings of The 2019 World Wide Web Conference, pages 148–153, 2019.
[19] Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. Botrgcn: Twitter bot detection with relational graph convolutionalnetworks. In Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pages 236–239, 2021.
[20] Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo.Heterogeneity-aware twitter bot detection with relational graph transformers. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 3977– 3985, 2022.
[21] Kristin Potter, Hans Hagen, Andreas Kerren, and Peter Dannenmann. Methods for presenting statistical information: The box plot. In VLUDS, pages 97–106, 2006.
[22] DonaldJ Berndt and James Clifford.Using dynamic time warping to find patterns in time series.InKDDworkshop, volume 10, pages 359–370. Seattle, WA, USA:, 1994.
[23] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 947–956, 2009.
[24] Wei Bao, Jun Yue, and Yulei Rao.A deep learning framework for financial time series using stackedautoencoders and long-short term memory. PloS one, 12(7):e0180944, 2017.
[25] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, GuillaumeAndroz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. Journal of Machine Learning Research, 21(118):1–6, 2020.
[26] David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. AMS math challenges lecture, 1(2000):32, 2000.
[27] Jun Wu, Patrick PC Lee, Qi Li, Lujia Pan, and Jianfeng Zhang. Cellpad: Detecting performance anomalies in cellular networks via regression analysis. In 2018 IFIP Networking Conference (IFIP Networking) and Workshops, pages 1–9. IEEE, 2018.
[28] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. IEEE Intelligent Systems and their applications, 13(4):18–28, 1998.
[29] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. Logistic regression. Springer, 2002.
[30] Fionn Murtagh. Multilayer perceptrons for classification and regression. Neurocomputing, 2(5-6):183–197, 1991.
[31] Leo Breiman. Random forests. Machine learning, 45:5–32, 2001.
[32] Fabian Pedregosa, Gae¨l Varoquaux, Alexandre Gramfort, Vincent Michel,Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, RonWeiss, Vincent Dubourg, etal. Scikit-learn: Machine learning in python. the Journal of machine Learning research, 12:2825–2830, 2011.

AUTHORS

Jun Wu holds a Master's degree in computer technology from Tsinghua University and is currently pursuing a Master's degree in Computer Science (AI-Track) at the Georgia Institute of Technology. Her research interests include machine learning, graph learning, anomaly detection, and their application in social networks and biomedical computing.

XueSong Ye received Bachelor's Degree from the Chengdu University of Information Technology in Electronic and Information Engineering. He is pursuing his Master's Degree in Information Science from Trine University. His research interests include machine learning, health informatics.

Chengjie Mou received Master Degree from Sun Yat-sen University in software engineering. Currently, he is pursuing his Master Degree in Information Studies from Trine University. His research interests include named-entity recognition, text matching, graph-learning.