Revealing and Detecting Malicious Retweeter Groups

Nguyen Vo[†], Kyumin Lee[*], Cheng Cao[§], Thanh Tran[†], Hongkyu Choi[†]
[†]Department of Computer Science, Utah State University, Logan, UT, USA
[*]Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA, USA
[§]Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA
{nguyenvo,thanh.tran,hongkyu.choi}@aggiemail.usu.edu, kmlee@wpi.edu, chengcao@cse.tamu.edu

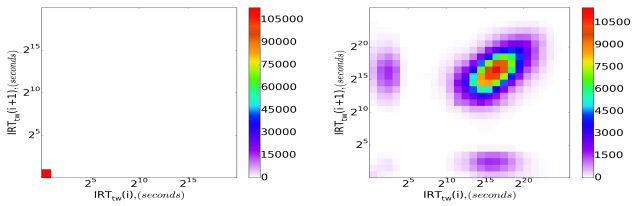
***Abstract*—Retweeting/sharing action has enabled information to be cascaded to distant nodes on social network. Unfortunately, malicious users as a group have taken advantage of the retweeting function with coordinated behavior to falsely distort the volume of specific keywords, topics or URLs for promotional purposes (e.g., spreading fake news, and increasing public visibility of products or services). Unfortunately, little is known about their retweeting behavior as a group and how to detect them based on group-based signals. To fill the gap, in this paper, we (i) propose Attractor+ algorithm to extract retweeter groups, members of each of which have similar retweeting behavior; (ii) analyze underlying characteristics of malicious and legitimate retweeter groups; (iii) propose group-based features to catch synchronized and coordinated behavior; and build a predictor to classify if a group is malicious. Experimental results show that our proposed method outperformed existing approaches.**

I. Introduction

Retweeting or sharing posts (e.g., messages, photos, videos) is one of popular actions on online social networking sites such as Twitter and Facebook. [1] found that a quarter of tweets were retweeted by friends. According to Sysomos, 6% of tweets is retweets, and 92% of retweets happened within the first hour after the original tweets were posted [2], showing how quickly people responded to spread information. A retweeting or sharing function has provided users and society with various benefits like spreading news and information in emergency situations (e.g., Boston Marathon bombing, Hurricane Sandy), supporting politicians (e.g., presidential candidates), creating grass-roots campaigns, and so on.

However, some users and organizations have begun boosting the number of retweets in inorganic ways such as purchasing retweeting services (e.g., traffup.net/retweets), hiring crowd workers from crowdsourcing platforms (e.g., fiverr.com and seoclerk.com), and deploying bots.

In this paper, we call these paid retweeters, spammers and bots *malicious retweeters*. A group of malicious retweeters, who often retweet paid or intentionally targeted tweets together, is called a *malicious retweeter group*. These malicious retweeter groups create wrong impression about certain news, topics and products by boosting the volume of retweets and eventually degrade quality and trust of information systems.



(a) Malicious group (b) Legitimate group

Fig. 1. Comparison between malicious groups and legitimate groups in terms of their inter-retweeting times. The x and y axes are ith and $(i+1)$th inter-retweeting times corresponding to each original tweet tw, respectively.

Recently, researchers made attempts to detect individual spammers based on their retweeting activity [3], [4] or correlated bots [5]. However, these approaches may not work for malicious retweeter groups, each of which, has coordinated and synchronized retweeting behavior.

In this paper, we aim to explore group-based characteristics and build a framework to identify malicious retweeter groups and to complement individual malicious user detection methods. Prior work utilized collective signals by doing URL-based and content-based detections for other problems [6], [7], [8], [9]. However, detecting synchronized malicious retweeter groups was not studied yet. To fill this gap, in this paper, we (1) propose Attractor+ to extract retweeter groups; (2) study latent properties of malicious and legitimate retweeter groups; and (3) build a malicious retweeter group detector based on temporal and content-based features. An interesting feature is aggregated inter-retweeting times (IRT) inspired by [10]. By gathering IRT of retweeters in a retweeter group (e.g., the gap between two consecutive retweets of each target tweet) and plotting IRT pairs in log-log scale, we observe that malicious retweeter groups usually have short and similar IRT (see Fig. 1(a)), whereas legitimate retweeter groups have longer and diffuse IRT (see Fig.1(b)). Interestingly, malicious and legitimate groups in the example shown in Figure 1 had 119,080 and 642,670 respectively. It contradicts the intuition that malicious retweeters tend to retweet more to promote products/services. Our main contributions are as follows:

ASONAM '17, July 31-August 03, 2017, Sydney, Australia

ACM ISBN 978-1-4503-4993-2/17/07...$15.00
http://dx.doi.org/10.1145/3110025.3110068

- First, we proposed Attractor+ to extract retweeter groups, users in each of which had similar retweeting behavior.
- Second, we explored synchronized and coordinated retweeting behavior of malicious retweeter groups in terms of temporal and content-based properties. Based on the analysis, we proposed novel group-based features.
- Finally, we built detectors based on the group-based features. Our model achieved 0.914 AUC and 91% accuracy, and outperformed other existing detection approaches.

II. Related Work

Retweeting action is a means of disseminating information across social network. Many recent efforts were made to investigate this powerful function. Researchers studied to understand retweeting behavior [1], [11], derived the likelihood that a tweet will be retweeted [12], [13], [14], and predicted how many times a tweet will be retweeted [15]. Another direction was to target right strangers for propagating content when they were requested [16].

Unethical and malicious users took advantage of the retweeting function in inorganic ways to boost popularity of specific content on social network, leading to deterioration of other users' experience. To tackle the problem, some methods were proposed. [3] ranked users based on their retweeting similarity and following-follower graph. [4] extracted mixtures of retweeters and their tweets to detect spammers. However, these prior works neglected group-based signals of retweeters. Some researchers exploited collective signals to categorize retweet threads [9] and to detect fraudsters [7], [17]. The goal of these works were different from ours since we focused on detecting malicious retweeter groups. Besides, researchers studied how to detect bots in social networking sites [18], [5].

Community detection algorithms [19] were adopted to extract disjointed connected subgraphs. Some previous works employed these algorithms to detect URL and content-based campaigns on Twitter [6], [8]. But their research problems are different from ours. As retweeting brings people into a specific conversation [11], we aim to explore malicious retweeter groups and find discriminating patterns between malicious retweeter groups and legitimate retweeter groups.

III. Our Proposed Approach

In this section, we describe our approach to detect a *malicious retweeter group*, users of which often retweeted paid or intentionally targeted tweets together. A malicious retweeter group detection problem is divided to two sub-problems: (a) how can we extract retweeter groups?; and (b) how can we determine whether the extracted group is malicious.

To solve these research problems, we propose an approach consisting of the four steps: (i) measure pair-wise similarity of users based on their retweeting behavior; (ii) build undirected weighted graph; (iii) extract potential retweeter groups (i.e., subgraphs) from the user graph, and further decompose each potential retweeter group to its subgraphs based on other properties (e.g., similar user names or email addresses), considering each of the subgraphs as a retweeter group; (iv) extract group features from each retweeter group and build a malicious retweeter group detector. The purpose of (1)∼(3) and (4) is to extract retweeter groups and predict whether each retweeter group is malicious or not, respectively.

A. Building a User Graph Based on Retweet Behavior

For each user u_i in our dataset, $\text{RT}(u_i)$ denotes the set of retweets of u_i. A tuple $(|\text{RT}(u_i)|, |\text{RT}(u_j)|, |\text{RT}(u_i) \cap \text{RT}(u_j)|)$ indicates the number of retweets of user u_i and user u_j, and their common retweets, respectively. We measure retweeting similarity between u_i and u_j by proposing mixed-similarity measure as follows:

$$sim(u_i, u_j) = \alpha \cdot cosine(u_i, u_j) + (1 - \alpha) \cdot overlap(u_i, u_j) \quad (1)$$

where $cosine(u_i, u_j)$ and $overlap(u_i, u_j)$ are defined below.

$$cosine(u_i, u_j) = |\text{RT}(u_i) \cap \text{RT}(u_j)| / \sqrt{|\text{RT}(u_i)| \times |\text{RT}(u_j)|}$$
$$overlap(u_i, u_j) = |\text{RT}(u_i) \cap \text{RT}(u_j)| / min(|\text{RT}(u_i)|, |\text{RT}(u_j)|)$$

By using Equation 1, we build a weighted undirected graph $G(V, E)$, where V is a set of users and $E = \{(u_i, u_j) | u_i, u_j \in V, sim(u_i, u_j) > \tau\}$. We only add an edge between users u_i and u_j if their pair-wise similarity score is greater than τ.

B. Extracting retweeter groups

Given a user/retweeter graph, a common way to extract groups of users is to run a community detection algorithm. In prior work, community detection methods are adopted to extract social campaigns [6], [8]. But, extracting near-clique subgraphs [8] neglected the weight of edges and took long running time. Louvain algorithm [19] has low-cost computation but its immediate subgraphs may not reflect underlying hierarchical structure of the original graph [21]. Due to these drawbacks, we propose *Attractor+*, a modified version of the Attractor algorithm [20], which examine the changes of distances among nodes over time. Attractor+ algorithm consists of four phases: (i) initialize distance of directly linked nodes (i.e., retweeters); (ii) perform dynamic interactions to update distances; (iii) extract potential retweeter groups from the graph; and (iv) if it is necessary, further decompose each potential retweeter group to its subgraphs based on other properties (e.g., similar user names or email addresses), considering each of the subgraphs as a retweeter group. In the first phase, Attractor+ algorithm initializes distance of edge (u, v) as follows: $d(u, v) = 1 - sim(u, v)$, where $sim(u, v)$ is the mixed similarity described in Equation 1.

In the second phase, Attractor+ measures dynamic interactions including direct linked interaction (DI), common interaction (CI) and exclusive interaction (EI). DI measures influence of directly linked nodes and is defined as follows.

$$DI(u, v) = sin(1 - d(u, v))/deg(u) + sin(1 - d(u, v))/deg(v)$$

, where $sin()$ is sine function and $deg()$ is node degree.

$CI(u, v)$ measures influence of common neighbors of u and v, denoted as $CN(u, v)$, and is equal to following expression:

$$\sum_{c \in CN(u,v)} \left(\frac{(1-d(v,c)) \cdot sin(1-d(u,c))}{deg(u)} + \frac{(1-d(u,c)) \cdot sin(1-d(v,c))}{deg(v)} \right)$$

$EI(u, v)$ measures influence of exclusive neighbors and is equal to following expression:

$\sum_{x \in EN(u)} \frac{\rho(x,v) \cdot sin(1-d(x,u))}{deg(u)} + \sum_{y \in EN(v)} \frac{\rho(y,u) \cdot sin(1-d(y,v))}{deg(v)}$

, where EN(u) is a set of exclusive neighbors of u, and $\rho(x, v)$ indicates influence of x on the $d(u, v)$. Given an input threshold $\lambda \in [0; 1]$, $\rho(x, v)$ depends on $\vartheta(x, v)$, the similarity of unconnected nodes x and v. If $\vartheta(x, v) \geq \lambda$, $\rho(x, v) = \vartheta(x, v)$ else $\rho(x, v) = \vartheta(x, v) - \lambda$.

We measure the similarity of x and v as follows:

$$\vartheta(x, v) = \frac{\sum_{c \in CN(x,v)} (1-d(x,c)+1-d(v,c))}{\sum_{k \in Neb(x)} (1-d(x,k)) + \sum_{l \in Neb(v)} (1-d(v,l))}$$

, where $Neb(x)$ is x's neighbors. After computing DI, CI, EI for edge (u, v), new distance at timestamp $t + 1$ is updated:

$$d(u, v)_{t+1} = d(u, v)_t - DI(u, v) - CI(u, v) - EI(u, v)$$

Attractor+ algorithm is looped until every edge is converged (e.g., its distance becomes either 0 or 1).

In the third phase, Attractor+ removes edges with distance equal to 1, and extracts connected components, each of which is a potential retweeter group.

To decide if it is necessary to further decompose a connected component, Attractor+ measures the density of a connected component and pair-wise screen name similarity. In particular, given a potential retweeter group, we build another graph, in which nodes are *screen names* of users in the group and the weight of each edge (u, v) is $\frac{EditDistance(u,v)}{max(|u|,|v|)}$. First, we remove pairs of screen names with Edit Distance similarity larger than a threshold ρ_1. Then, if density of the new graph is not larger than ρ_2, this group will be further decomposed by the decomposition algorithm presented in Algorithm 1. By varying ρ_1 and ρ_2, Attractor+ extracts retweeter groups each of which has a high density and/or similar screen names. According to [22], Twitter accounts having similar screen names may originate from the same person or organization who created them to spread information. The input of Algorithm 1 is a potential retweeter group which requires further decomposition. The idea of this algorithm is to choose seed nodes with the highest degree and perform breath first search to find neighbors, which have smaller or equal degree with the seed nodes. Each output subgraph is a retweeter group.

Algorithm 1 Decomposition algorithm

```
1: function DecomposeCommunity(V, E)
2:     Sorting degree of vertices decreasingly; S = ∅
3:     for ν ∈ V do: ν.seen = 0
4:     for each vertex top in sorted list and top.seen ≠ 1 do
5:         g = ∅ ; Q = {top}                          ▷ Initialize Queue
6:         while Q is not empty do
7:             u = Q.pop() ; u.seen = 1; g = g ∪ {u}
8:             for v ∈ Neb(u) and v.seen ≠ 1 and d_v ≤ d_u do
9:                 Q = Q ∪ {v} ; v.seen = 1
10:        S = S ∪ {g}
11:    return S
```

C. Identifying Malicious Groups

In the last step, we build predictive models based on 26 group-based features to detect malicious retweeter groups. These features are extracted from each retweeter group $RTG(V)$, where V is the set of retweeters/nodes in the group. $|V|$ indicates the number of retweeters of a group.

Retweet Related Features:

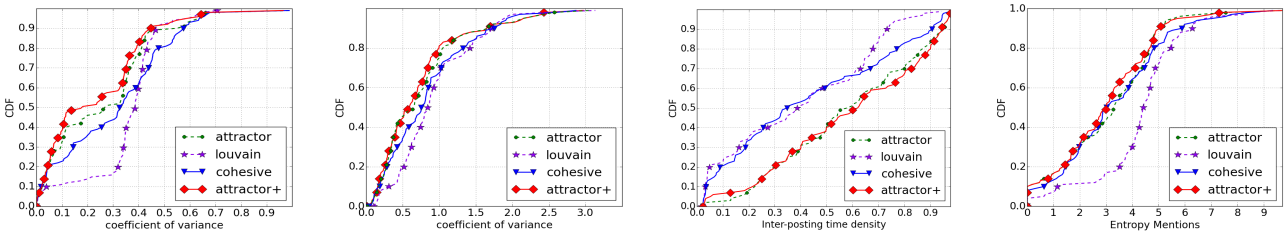
- Retweeted user related features (2 features): Retweeted users are users whose tweets were retweeted by $RTG(V)$. First, we computed |followers|/|followees| for each retweeted user. Then, we find mean value of their ratios. In addition, we counted the number of verified distinct retweeted users.
- Average similarity score of pairs of retweeters in a retweeter group measured by Equation 1.
- Inter-posting time density: For every retweeter $r_i \in V$, we collected his tweets including retweets, and extracted posted time of the tweets. Then, we merged all the posted times of the retweeters, and sorted them increasingly, resulting in a sorted list $\mathbf{P} = [p_1, p_2, p_3, ..., p_k]$. Next, we measured inter-posting times Δ of $\mathbf{P}$ (i.e., $\Delta = [p_2 - p_1, p_3 - p_2, ..., p_k - p_{k-1}]$). Then, each pair of consecutive inter-posting times (e.g., (Δ_1, Δ_2), (Δ_2, Δ_3)) was logarithmically binned into a square grid in two-dimensional space (e.g., ($\log_2 \Delta_i$, $\log_2 \Delta_{i+1}$)). We counted how many pairs were in each grid cell and stored it in $\mathbf{PT} \in \mathbb{R}^{m \times m}$. IPTDensity is defined as follows:

$$IPTDensity(V) = \max_{i,j} (\mathbf{PT}_{ij} \ / \ \sum_{i=1,j=1}^{m} \mathbf{PT}_{ij})$$

- Retweeter-centered retweeting time dispersion (3 features): For each retweeter $r_i \in V$, we extract a list of retweeting times and measure the standard deviation of this list. From $|V|$ standard deviations, we measure three features – the mean, standard deviation and coefficient of variance.
- Target-centered retweeting time dispersion (2 features): First, we collected original tweets tw (also called target tweet) retweeted by retweeters $r_i \in V$. With each tw, we extracted a list L_{tw} of retweeting times of retweets from all $r_i \in V$. Then, we measured coefficient of variance of L_{tw}. Now, we have a list of coefficient of variances, each of which was obtained from each target tweet tw. Then we computed two features, the median and standard deviation of the list.
- Target-centered inter-retweeting time density: We use L_{tw}, defined in the previous feature, for each original tweet tw . Then, we sort L_{tw} in the ascending order and measure inter-retweeting times. Now, we have a list $\mathbf{L}$ of inter-retweeting times for tw. Then, each pair of consecutive inter-retweeting times in $\mathbf{L}$ was logarithmically binned into a square grid in 2D space. We counted how many pairs were in each grid cell, stored it in $\mathbf{R} \in \mathbb{R}^{m \times m}$ and derived following feature:

$$TIRTDensity(V) = \max_{i,j} (\mathbf{R}_{ij} \ / \ \sum_{i=1,j=1}^{m} \mathbf{R}_{ij})$$

- Coefficient of variance of response times: First, we measure the median of response times for each retweeter $r_i \in V$. Response time means the gap between retweeting time and original tweet's posted time to see how long it took r_i to retweet the original tweet. Given a list of $|V|$ medians, we get the coefficient of variance of it as a feature.
- Response time density: This feature is used to capture what range of response times that retweeters $r_i \in V$ usually have. First, extract response times from all $r_i \in V$. Then, merge and sort the response times in the ascending order. Next, each pair



(a) Account Creation Date (b) Retweeting Time (c) Inter-posting time Density (d) Entropy mention

Fig. 2. Comparison of four retweeter group extraction algorithms in four properties which reflect synchronized behavior.

of consecutive response times is logarithmically binned into a square grid in 2D space. Then, we count how many pairs are in each grid cell and store it in a matrix $\mathbf{RE} \in \mathbb{R}^{m \times m}$. Finally, we compute the following feature:

$$REDensity(V) = 1/(i+j) \cdot \max_{i,j} \left(\mathbf{RE}_{ij} \;/\; \sum_{i=1,j=1}^{m} \mathbf{RE}_{i,j}\right)$$

, where i and j are indexes of a grid cell which has the largest count. If most retweeters in $RTG(V)$ quickly retweet original tweets, most response times will be mapped to the left and low grid cell, and $REDensity$ will be large.

• Pairwise retweeting time similarity: For every pair of retweeters (r_i, r_j) where $sim(r_i, r_j) > \tau$, we extract their commonly retweeted target tweets $CROT(r_i, r_j)$. For each tweet tw of $CROT(r_i, r_j)$, we find retweeting time difference: $\text{diffT}_{tw}(r_i, r_j) = \left(rtTime_{r_i}(tw) - rtTime_{r_j}(tw)\right)^2$, where $rtTime_r(tw)$ denotes the time when retweeter r retweeted an original tweet tw. Finally, we compute their retweeting time similarity as follows:

$$PRTSim(r_i, r_j) = 1/\left(1 + \sqrt{\sum_{tw \in CROT(r_i, r_j)} \text{diffT}_{tw}(r_i, r_j)}\right)$$

The larger $PRTSim(r_i, r_j)$ is, the more similar their retweeting times are. Finally, we find the median $PRTSim$ of all the pairs/all edges in the group and use it as a feature.

Retweeter Related Features:

• Standard deviation of retweeters' registration dates, mean and coefficient of variance of retweeters' |followers| (2 features), mean of |followers| / |followees| of retweeters

• Bi-directional followers: For each retweeter r_i, we get $\frac{|followers(r_i) \cap friends(r_i)|}{|followers(r_i)|}$. Then, average retweeter's value.

• Average length of retweeters' screen names, |retweeters who have URL in their bios|/|V|, |retweeters who have their bios|/|V|, and $|digits|$ *in screen names*/|V| (4 features).

• Average edit distance similarity: We find Edit Distance similarity of screen names u and v and derive average value based on the number of edges in $RTG(V)$.

URL, Hashtag and Mention based Features: Mean of |URLs|, |mentions| and |hashtags| (3 features): We collected each retweeter's all tweets and then derived the three features.

IV. Experiments

A. Data Collection and Building a User Graph

We collected 1.6 billion tweets from Twitter by using Twitter Streaming API between November 2014 and November 2015. Since we were interested in users' retweeting behavior, we discarded less active users who had posted less than 5 retweets, resulting in 21 million users. We removed a pair of users if the number of their commonly retweeted tweets was less than four times because too few common retweets does not reliably reflect similarity of their retweeting behaviors. After the filtering, 57,628,118 pairs (of 795,381 distinct users) were remained. Then, we computed mixed similarity of all pairs by using Equation 1. It took only 3 hours to process 57M pairs under our 40 cores-based Hadoop system. Ideally, we should vary α to find the optimal value in the mixed similarity measure. But, practically it's not easy to validate which α value returns optimal results. Therefore, we chose $\alpha = 0.5$ to balance contribution of Cosine and Overlap similarity.

When we build a user graph based on users' retweeting behavior, the size of graph depends on τ. If τ is too small, the user graph will be too big, requiring longer computation time, and extracted retweeter groups will be less similar and less interesting. Thus, we chose $\tau = 0.3$, keeping 2% of all the pairs. Finally, we built a user graph $G(V, E)$ where $|V| = 199,981$ and $|E| = 1,159,674$. We also collected followees, followers and recently posted 1,000 tweets of each user $\in V$.

B. Extracting Retweeter Groups

Now we turn to extract retweeter groups by using each of the four subgraph detection algorithms to see how their extracted groups are similar or different, and show the effectiveness of our Attractor+. The four algorithms are Cohesive, Louvain, Attractor and Attractor+. We use $\lambda = 0.5$ in both Attractor and Attractor+ like [20]. In Attractor+, we varied $\rho_1 \in [0, 1]$ and $\rho_2 \in [0, 0.1]$, and found that $\rho_1 = 0.3$ and $\rho_2 = 0.03$ returned the best result (i.e., each group had similar interest and purpose).

Now, we compare the largest 100 groups extracted by each algorithm. The standard metrics (e.g., NMI, ARI, Purity) to evaluate community detection algorithms do not reflect coordinated behavior of retweeters. Instead, in this paper, our goal is to extract retweeter groups, members in each of which have coordinated behavior. Therefore, we used four properties to understand which method found groups containing more synchronized behaviors. Figure 2(a) shows CDF of coefficient of variance of retweeter account creation dates. We measured a coefficient of variance for each group and plotted the Figure. Attractor+ achieved smaller coefficient of variance than the others. Note that a small coefficient of variance means retweeters in the same group created their accounts in similar date or small range of the time. In Figure 2(b), retweeter groups extracted by Attractor+ had smaller coefficient of variance in

retweeter-centered retweeting time dispersion described in the previous section. It means retweeters in each group had similar retweeting time. In Figure 2(c), retweeter groups extracted by Attractor+ and Attractor had larger IPTDensity. A large IPTDensity means that there is high concentration on inter-posting times of retweeters in each group, an indication of synchronized posting behavior. In Figure 2(d), Attractor+ and Attractor had smaller entropy of mentioned users in tweets posted by retweeters in each group. It means retweeters in each group mentioned similar Twitter accounts. In the analysis and comparison, Attractor+ found user groups which had more synchronized behaviors than the other methods. When we analyzed the largest 1,000 groups, we got consistent results.

C. The Ground Truth

By using Attractor+, we extracted 43,012 retweeter groups. Labeling all the retweeter groups is not feasible because labelers have to check each group's all retweeters. Instead, we sampled 1,000 retweeter groups, following the same group size distribution of the 43,012 retweeter groups to avoid potential bias. In particular, we used Sturges equation [23] to choose the number of bins approximately equal to $17 \approx 1 + \log_2(43012)$, and then selected 1,000 retweeter groups.

Labeling a retweeter group was more sophisticated and required more careful examination than labeling individual user as a spammer or not. Therefore, we hired 3 full-time labelers rather than randomly hiring workers from crowd-sourcing sites such as Amazon MTurk. The labelers spent one month for labeling 1,000 groups on a scale from 1 (strongly legitimate) to 5 (strongly malicious). In the labeling process, they referred to Twitter's terms of service to determine whether a group is malicious or not. In particular, they looked at retweeters' timelines and their posting/retweeting behavior in each retweeter group. The labelers did not have any knowledge about what features are used in our paper, nor the authors did not influence their rating criteria. After collecting their labeling results, we averaged their ratings for each group. If an average rating for a group was larger than 3, we finally labeled it as a malicious group. If a rating was less than 3, we labeled it as a legitimate group. If the rating was 3 (neutral), they re-checked the group, and made decision together. Their Kappa value was 0.350 which is fair agreement between labelers [24]. In particular, they had same rating for 427 groups, and two of them had the same rating scores for 511 retweeter groups which have max, min and mean standard deviation of three rating scores are 1.414, 0.471, 0.529 respectively. Finally, 769 out of the 1,000 groups (76.9%) were labeled as malicious. It makes sense because doing synchronized or coordinated behavior is not usually normal behavior.

D. Identifying Malicious Retweeter Groups

First, we extracted 26 feature values from each of the 1,000 groups. All pairs of the features has Pearson coefficient less than 0.5. When ranking features with χ^2 values, we observe that the most important features were mostly related to temporal behaviors of retweeters (e.g., IPTDensity).

Method	AUC	F1	Precision	Recall
[17] (**)	0.382	0.419	1.000	0.265
[6] (*)	0.894	0.843	0.845	0.848
Our approach	**0.914**	**0.874**	**0.876**	**0.878**

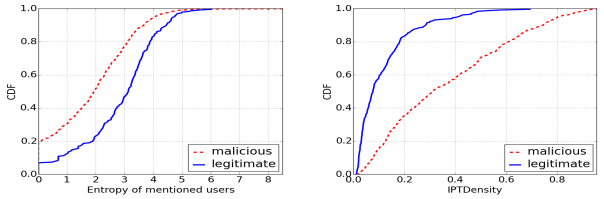
TABLE I
PERFORMANCE OF OUR APPROACH AND TWO BASELINES. WILCOXON TEST CONFIRMS THE SUPERIORITY OF OUR APPROACH OVER [6] AND [17] METHOD IS SIGNIFICANT. (*) P-VALUE<0.05, (**) P-VALUE<0.001

Method	Accuracy	FPR	FNR
Baseline 1 [17] (***)	24.2%	0.242	0.757
Baseline 2 [3] (***)	46.3%	0.014	0.704
Baseline 3 [25] (**)	82.4%	0.305	0.172
Baseline 4 [6] (*)	86.1%	0.241	0.101
Our approach	**91.0%**	0.197	0.050

TABLE II
OUR APPROACH VS. FIVE BASELINES. WILCOXON TESTS CONFIRMED THAT THE SUPERIORITY OF OUR METHOD OVER BASELINES IS SIGNIFICANT. (*) P< 0.05, (**) P<0.01, (***) P< 0.001

We tried various machine learning algorithms and found out that XGBoost classifier performed the best based on our group-based features. Next, we compare our best classifier with two group-based baselines [6], [17]. In [6], we extracted 9 group-based features and built a predictive model. [17] used two thresholds (i.e. number of retweeters in a group and median of absolute time intervals) to identify malicious campaigns. We optimized two thresholds based on training set and applied them to test set. We conducted 10-fold cross-validation for the three methods. Table I presents the results of XGBoost and two baselines. Our approach achieved 0.914 AUC, significantly outperforming two baselines.

Detection of Individual Malicious Retweeters. Next, we compare our approach against existing malicious retweeter detection approaches, focusing on detecting individual malicious retweeters. To get the ground-truth of each user in the 1,000 retweeter groups, we assumed that users in malicious groups are malicious retweeters and users in legitimate groups are legitimate retweeters. With a sampling approach, we verified that the assumption was correct. The 1,000 retweeter groups consisted of 7,505 malicious and 2,809 legitimate retweeters. We compared our approach with the following five baselines.
Baseline 1: [17] proposed validation methods to identify spammers after extracting potentially malicious campaigns. Based on their methods, we check if an URL is malicious. If any tweet of a poster contains malicious URLs, we consider the poster as malicious.
Baseline 2: [3] proposed a PageRank-based method in which they built a weighted directed graph, where each node was a retweeter and an edge (u, v) was formed if user u followed user v. Edge weight of (u, v) was based on the similarity of u and v's retweet behavior. Given initial seed nodes, it computes a spam score by using PageRank. Based on a threshold, it decides which users are malicious retweeters. We found optimal parameters in a training set and applied to a test set.
Baseline 3: [25] proposed a predictive model based on each user's behavior. Based on their proposed features, we built a XGBoost-based classifier.
Baseline 4: We consider users in detected malicious groups



(a) Entropy of mentions (b) IPTDensity

Fig. 3. Malicious vs. legitimate groups in two properties. One-sided Mann-Whitney U-test confirmed the difference is significant (p-value$<$ 0.001).

by [6] as malicious and users in legitimate ones as organic.

We fairly conducted 10-fold cross validation for all the methods. We computed accuracy, false positive rate (FPR) and false negative rate (FNR). As shown in Table II, our approach significantly outperformed the five baselines, achieving 91.0% accuracy, 0.197 FPR and 0.050 FNR.

Robustness of our approach. We created another 9 datasets from the original 1,000 groups by reducing the number of malicious groups. In particular, we randomly selected 90% to 10% of the 769 malicious retweeter groups, decreasing 10% for each dataset and keeping 231 legitimate retweeter groups. Then we built and tested our model in each of 9 datasets. With 95% confidence interval, our approach achieved stable results and outperformed all baselines, especially [6] and [25] (p-value$<$0.01). In particular, the AUC and F1 of our malicious retweeter group detection were $90.9\% \pm 0.85\%$, and $85.3\% \pm 0.88\%$ respectively. The accuracy, FPR and FNR of detecting individual malicious retweeters were $89.8\% \pm 0.93\%$, $11.6\% \pm 2.98\%$, and $12.3\% \pm 4.08\%$, respectively.

In addition, the labelers labeled the largest 1,000 retweeter groups among 43,012 groups. Our approach in the dataset achieved even better results – 0.954 AUC and 95.2% accuracy. These results indicate the robustness of our proposed approach.

V. Malicious Groups vs. Legitimate Groups

In this section, we conduct analysis to understand behavioral differences between malicious and legitimate retweeter groups.

RQ#1: What content did malicious retweeting groups post? We analyzed all tweets of retweeters in 1000 groups and measured entropy of hashtags, URLs and mentions. Figure 3(a) shows that entropy value of malicious groups was smaller than that of legitimate ones, indicating that retweeters in each malicious group mentioned similar accounts to promote them. We also observed similar patterns in entropy of hashtags and URLs.

RQ#2: Did malicious retweeters have synchronized behavior?

Figure 3(b) shows IPTDensity of two types of retweeter groups. Malicious groups had much larger IPTDensity than legitimate ones (0.364 vs. 0.119), indicating more synchronized behavior. Malicious groups also had larger TIRTDensity than legitimate groups (0.954 vs. 0.874). Figures 1(a) and 1(b) show heat maps of two groups with their TIRTDensity. The malicious group's IRT for the same target tweets were 1$\sim$2 seconds, leading to 0.99 TIRTDensity, whereas the legitimate group's IRT were diffuse, leading to 0.028 TIRTDensity.

VI. Conclusion

In this paper, we proposed a framework to detect malicious retweeter groups. In particular, Attractor+ algorithm extracted retweeter groups, the members of each of which had similar retweeting behavior. Then, we proposed group-based features to detect malicious retweeter groups. Our method achieved 0.914 AUC in the malicious group detection and 91.0% accuracy in individual malicious retweeters detection.

Acknowledgment

This work was supported in part by NSF grant CNS-1553035. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

References

[1] Z. Yang, J. Guo, K. Cai, J. Tang, J. Li, L. Zhang, and Z. Su, "Understanding retweeting behaviors in social networks," in *CIKM*, 2010.

[2] Sysomos, "https://sysomos.com/inside-twitter/twitter-retweet-stats," in *Replies and Retweets on Twitter*, 2010.

[3] B. Liu, J. Luo, J. Cao, X. Ni, B. Liu, and X. Fu, "On crowd-retweeting spamming campaign in social networks," in *ICC*, 2016.

[4] Z. Qunyan, Z. Chi, C. Peng, Q. Weining, and Z. Aoying, "Detecting spamming groups in social media based on latent graph," in *ADC*, 2015.

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

[6] C. Cao, J. Caverlee, K. Lee, H. Ge, and J. Chung, "Organic or organized? exploring url sharing behavior," in *CIKM*, 2015.

[7] G. Maria, C. Despoina, S. Neil, B. Alex, F. Christos, and V. Athena, "Nd-sync: Detecting synchronized fraud activities," in *PAKDD*, 2015.

[8] K. Lee, J. Caverlee, Z. Cheng, and D. Z. Sui, "Campaign extraction from social media," *ACM Trans. Intell. Syst. Technol.*, 2014.

[9] L. K. G. Rumi, S. Tawan, "Entropy-based classification of'retweeting'activity on twitter," *arXiv:1106.0346*, 2011.

[10] A. Ferraz Costa, Y. Yamaguchi, A. Juci Machado Traina, C. Traina Jr, and C. Faloutsos, "Rsc: Mining and modeling temporal activity in social media," in *KDD*. ACM, 2015.

[11] Boyd, Danah, S. Golder, and G. Lotan, "Tweet, tweet, retweet: Conversational aspects of retweeting on twitter," in *HICSS*, 2010.

[12] N. Naveed, T. Gottron, J. Kunegis, and A. C. Alhadi, "Bad news travel fast: A content-based analysis of interestingness on twitter," in *WebSci*, 2011.

[13] S. Petrovi, "Rt to win! predicting message propagation in twitter," in *ICWSM*, 2011.

[14] B. Suh, L. Hong, P. Pirolli, and E. H. Chi, "Want to be retweeted? large scale analytics on factors impacting retweet in twitter network," in *SocialCom*, 2010.

[15] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in twitter," in *International World Wide Web Conference*, 2011.

[16] K. Lee, J. Mahmud, J. Chen, M. Zhou, and J. Nichols, "Who will retweet this? automatically identifying and engaging strangers on twitter to spread information," in *IUI*, 2014.

[17] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *SIGCOMM*. ACM, 2010.

[18] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *arXiv preprint arXiv:1407.5225*, 2014.

[19] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[20] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *KDD*, 2015.

[21] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[22] R. Zafarani and H. Liu, "Connecting users across social media sites: a behavioral-modeling approach," in *KDD*, 2013.

[23] D. V. Huntsberger, *Elements of statistical inference*. Allyn & Bacon, Inc., 1986.

[24] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, 1977.

[25] K. Lee, B. D. Eoff, and J. Caverlee, "Seven months with the devils: A long-term study of content polluters on twitter," in *ICWSM*, 2011.