Adaptive Graph-based algorithms for Spam Detection in Social Networks

AMIRA SOLIMAN, SARUNAS GIRDZIJAUSKAS

Stockholm

SCS/ICT/KTH

Adaptive Graph-based algorithms for Spam Detection in Social Networks

Amira Soliman and Sarunas Girdzijauskas

Royal Institute of Technology (KTH), Sweden
{aaeh, sarunasg}@kth.se

September 29, 2016

Abstract

As Online Social Networks (OSNs) continue to grow in popularity, a spam marketplace has emerged that includes services selling fraudulent accounts, as well as acts as nucleus of spammers who propagate large-scale spam campaigns. In the past years, researchers developed approaches to detect spam such as URL blacklisting, spam traps and even crowdsourcing for manual classification. Although previous work has shown the effectiveness of using statistical learning to detect spam, existing work employs supervised schemes that require labeled training data. In addition to the heavy training cost, it is difficult to obtain a comprehensive source of ground truth for measurement.

In contrast to existing work, in this paper we present a novel graph-based approach for spam detection. Our approach is unsupervised, hence it diminishes the need of labeled training data and training cost. Particularly, our approach can effectively detect the spam in large-scale OSNs by analyzing user behaviors using graph clustering technique. Moreover, our approach continuously updates detected communities to comply with dynamic OSNs where interactions and activities are evolving rapidly. Extensive experiments using Twitter datasets show that our approach is able to detect spam with accuracy 92.3%. Furthermore, our approach has false positive rate that is less than 0.3% that is less than half of the rate achieved by the state-of-the-art approaches.

1 Introduction

With the widespread usage of user generated content in Online Social Networks (OSNs), spam in these sites is explosively increasing and has become an effective vehicle for malware and illegal advertisement distributions. Furthermore, OSNs have also led to new methods of delivering spam, such as spammy apps, social bots, and fake accounts resulting in increasing social media spam to 355% in 2013 over 2012 [17]. Spam content not only pollutes the content contributed by normal users, resulting in bad user experiences, but also can mislead or even trap legitimate users. As the spam has been plaguing OSNs for more than a decade, researchers have analyzed different spam strategies to design mechanisms to combat the spam activities from different prospectives, including studying the redirection chains of embedded URLs [12, 20, 13], classifying the URL landing pages [20, 2, 6], analyzing textual content [7, 23, 21], as

well as analyzing different friendship graph properities of spammers against those of legitemate users [22, 7, 23, 21].

Spotting spammers is very challenging especially with the dynamic nature of social networks where activities and interactions among users evolve rapidly. Furthermore, the problem becomes more challenging due to the huge amount of data shared by users. The research community has produced a substantial number of mechanisms for automated spam detection using machine learning techniques based on binary classification. The design of such spam detection mechanisms in general is guided by the behavior dissimilarity exhibited by legitimate users than spammers. The central premise as proved in the existing work is that spammer behavior appears anomalous relative to normal user behavior along some features that could be extracted from textual content (i.e., content-based features such as number of URLs, Hashtags and mentions used per post) and OSN friendship graph (i.e., graph-based features that are calculated from the friendship graph such as local clustering coefficient and betweenness centrality).

However, all of the existing techniques rely on supervised binary classification methods [7, 21, 1, 25, 20]. Although the proposed binary classification methods succeed at detecting spam content, they implicitly require offline training with statistically suffcient and representative labeled training set of different user behaviors in order to achieve good detection coverage. This requirement itself is hard to satisfy, not to mention the difficulty of adapting to different behavior patterns that emerge in the future. Furthermore, the number of features required to discriminate spammers increases due to the diverse users activists in OSNs, the evolving spam patterns, as well as the limited the amount of labeled data. For example, Zhu et al. [25] use 1,680 different user activities in their supervised detection approach and Thomas et al. [20] train their URL spam filtering method using a sparse feature space with possible number of features up to 10^7. Additionally, binary classification methods result in false positive rate that could range between 5.7% and 0.8% [15, 1] resulting in some legitimate users are identified as spammers and get disconnected from the network. Particularly, derived from the remark that spammers hijack trending topics and include many URLs in their posts, content-based classification methods distinguish spammers by the extensive use of URLs, hashtags and mentions. Consequently, legitimate users such as the official news channels that continuously broadcast posts with diverse topics containing URLs and hashtags of the trending topics are going to be classified as spammer.

To address these issues, in this paper we propose a novel unsupervised graph-based clustering technique for spam detection. Differently from existing work, our approach constructs a *user similarity graph* that is created by connecting users with edges having weights that quantify their behavioral similarity. The essence of our approach is to construct a user similarity graph that encodes within its topology a holistic view of all behavioral interactions and patterns of OSN users. Afterwards, our approach performs graph clustering by applying community detection on top of the newly created graph. In particular, we create a user-based feature vector to summarize both content and graph features associated with every user. Accordingly, the edges are created connecting users having weights equal to the cosine similarity of feature vectors of source and destination nodes [1]. Afterwards, our approach detects communities on top of similarity graph to identify different behavioral patterns existing in the social network, then spots the spam patterns among the detected ones by applying some lexical analysis. Spam detection using graph-based clustering not only diminishes the training cost, but also achieves low

[1]Users and nodes refer to the same meaning and are used interchangeably.

false positive rate. Graph-based clustering provides meaningful insights to the existing behavioral patterns, therefore, categorizes the existing patterns into more homogeneous and accurate clusters than binary splitting as illustrated in Figure 1. Hence, grouping users into multiple communities minimizes the chances of high false positive rates, specially for legitimate users with diverse and highly active behaviors such as news channel accounts. Clustering will group such accounts into a separate cluster with a closer distance to users having legitimate behavior pattern with diverse topics rather than the spam pattern that exhibit high URL and hashtags rate, yet in the same time has high similarity in the content. Hence, graph-based clustering provides more accurate results compared to binary classification without the need of the repetitive cost of maintaining up-to-date labeled training dataset.

However, centralized graph-based clustering technoques are not realistically scalable due to the huge number of users in current OSNs. Therefore, graph-based clustering algorithns must be developed as massively parallel clustering that eliminates the need of single centralized aggregation point. Even better, graph-based clustering can be implemented as fully decentralized solution to be applicable with currently emerging Decentralized Online Social Networks (DOSNs). DOSNs operate as distributed information management platforms on top of networks of trusted servers or P2P infrastructures [4]. Thus, DOSNs provide a privacy preserving alternative to current OSNs, where users have full control of their data. Accordingly, in our approach we allow every node to independently process its data and only communicate with its direct neighbors. Additionally, our approach adaptively updates similarity connections among and the detected communities based on the newly received information integrated with the previously known without the need of recomputing from scratch. Hence, our approach is capable of monitoring the behavioral changes and dynamically adapts to the evolving social activities and interactions among users. We have performed experiments, using Twitter datasets, to show the effectiveness of our proposed approach. The results show that our approach provides more accurate spam detection rate with accuracy up to 92.3% and false positive rate less than 0.3%. Thus, our approach outperforms the state-of-the-art techniques not only in plain performance figures, but also by removing the need of labeled data and offline training effort (since our approach is unsupervised) as well as removing the scalability issues due to the fully decentralized and distributed nature of the algorithm.

Accordingly, our work offers the following contributions to the problem of spam detection:

- Unsupervised spam detection approach that requires no a priori labeling while maintaining low false positive rate,
- A novel graph-based spam detection technique that detects spam using graph clustering on top of a constructed user similarity graph which encodes user behavioral patterns within its topology,
- Adaptive similarity-based community detection that evolves with respect to the behavioral changes of the users,
- Community detection based classification that is better performing than binary classification,
- All of the above contributions are performed in purely distributed and decentralized manner.

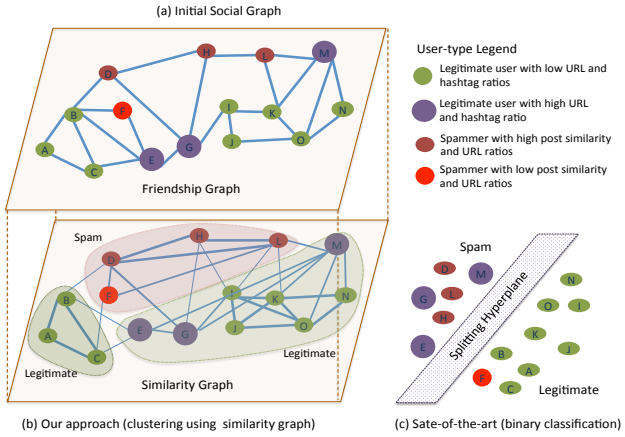


Figure 1: Similarity-based clustering vs. Binary classification. (a) Initial social graph of OSN users having different behavioral patterns. (b) Our approach creates similarity graph and extracts communities that group users with similar behaviors. (c) Binary classification organizes all users in feature space to find the best splitting hyperplane.

The remainder of this paper is structured as follows. In Section 2 we list the features used for spam detection, whereas, in Section 3 we illustrate the core of our proposed approach starting with the algorithms used to construct similarity graph followed by algorithms for community detection and community structure adaptation. Furthermore, in Section 3 we detail the lexical analysis method adopted to indicate the spammers communities among the detected ones . In Section 4 we present evaluation of our approach. Finally, Section 5 shows the related work, then Section 6 concludes the paper.

2 Spam Detection Features

In this section, we first briefly describe the features used in our framework to compute user-based feature vectors. We organize the features in two categories: graph-based and content-based features.

2.1 Graph-based features

In this part we utilize the original social friendship graph connecting users. We consider the social network as a directed graph G = (V, E), where V is the set of nodes and E is the set of edges. $e_{ij} \in E$ denotes a relationship between nodes v_i and $v_j \in V$.

Definition 2.1. *Local Clustering Coefficient (LCC). We use the LCC instead of node's degree as spammers can gain more followers by purchasing them. A node's LCC [5] is the ratio between the number of existing links among its direct neighbors and the number of links that could possibly exist between them. This metric is used to quantify the extent to which the direct neighbors of a node are connected to each other. Given* $v_i \in V$*, let* $DF_i = \{v_j \in V | e_{ij} \in E\}$ *be the set of* v_i*'s direct friends.* LCC_i *represents the local clustering coefficient of* v_i*'s, and equals to:*

$$LCC(v_i) = \frac{|e_{jk} : v_j, v_k \in DF_i, e_{jk} \in E|}{|DF_i|(|DF_i| - 1)} \tag{1}$$

Due to decentralized nature of our approach, we assume that every node calculates its *LCC* locally by keeping track of two-hop neighbors (i.e., neighbors of the neighbors).

Definition 2.2. *Average Neighbors Clustering Coefficient (ANCC). ANCC metric is used to quantify the connectedness of the neighborhood of a node. Madden et al. [14] show that majority of OSN users are more skeptical regarding the acceptance of new friendship requests from strangers. Therefore, it is hard for spammers to have strongly connected neighborhood surrounding them. Thus, ANCCes of legitimate users are commonly higher than those of spammers. We define ANCC as follows:*

$$ANCC(v_i) = \frac{\sum_{v_j \in DF_i} LCC(v_j)}{|DF_i|} \tag{2}$$

2.2 Content-based features

A recent study [9] shows that most spam is generated using templates. Furthermore, spammers go for more complex templates such as finite-state machines to evade spam detection methods [3]. Although, finite-state machines increase the number of different spam posts that can be generated, all of the generated posts follow a structured content, for example [mentions + some text + URLs + hashtahs]. Hence, spam posts still have some words in common such as "look at this video" or "gain more", etc. Consequently, spam posts can be captured by measuring the posts similarity.

Definition 2.3. *Average Posts Similarity (APS). This feature leverages the similarity among the posts shared by a single user. We define post similarity using jaccard coefficient, such that for every post pair of a user, we divide the intersection (i.e., the number of shared words) by the total number of words in the post pair. Let P_i be the set of posts shared by v_i, and p_{jk} be the pair of two posts p_j and p_k in P_i. We define the average posts similarity of v_i as follows:*

$$APS(v_i) = \frac{1}{\binom{|P_i|}{2}} \sum_{p_{jk} \in P_i} \frac{p_j \cap p_k}{p_j \cup p_k} \tag{3}$$

Due to decentralized nature of our approach, we assume that posts are publicly available and can be collected by other nodes.

Definition 2.4. *Mentions Ratio (MR). Mentions have been intensively used by spammers to increase the visibility of their content, such that spammers add mentions to random users. Whereas, legitimate users interact with their friends, hence mentions can be used to detect spammers. Accordingly, we define MR as the number of posts containing mentions of users whom are not directly connected to that node divided by the total number of posts a user has. We define mention ratio as follows:*

$$MR(v_i) = \frac{|@username \in P_i, \exists username \notin DF_i|}{|P_i|} \tag{4}$$

Definition 2.5. *URL Ratio (UR). Spammers embed malicious URLs in their posts to direct the users to their websites. Thus, the percentage of posts containing URLs has been used as an effective indicator of spam accounts. We define UR as the the ratio*

of the number of posts containing a URL to the total number of posts a user has (i.e., $|URLs|/|P_i|$).

Definition 2.6. *Hashtags Ratio (HR). Hijacking popular topics in OSNs (i.e., trending topics) has been a widely adopted strategy among spammers to reach wider audience. Therefore, we define HR as the number of trending topics associated with user posts to the total number of posts (i.e., $|Hashtahs|/|P_i|$).*

3 Graph-based Spam Detection

In this section, we present the core of our approach. First, we illustrate the construction of user similarity graph, followed by the details of the local clustering algorithm. Afterwards, we present the quick community adaptation algorithm used for tracing the evolution of users' behavioral patterns represented in detected communities over time. Furthermore, we discuss the complexity of our framework and present the adopted lexical analysis approach to spot spammers communities among detected ones.

3.1 Similarity Graph Construction

The first step of our approach is to construct users similarity graph from the social graph. To build a massively parallel approach, we allow every node in the social graph to participate in similarity graph construction. Initially, every node starts by creating similarity edges among itself and its social neighbors. The edges are created by connecting any pair of nodes having cosine similarity of their feature vectors greater than specific threshold. So as, the weight of an edge connecting node *i* and node *j* equals to:

$$w(e_{ij}) = \frac{x_i.x_j}{||x_i||.||x_j||} \tag{5}$$

Where, x_i is the feature vector of node *i* and x_j is the feature vector of node *j*. Particularity, the feature vector of a node has the following values [LCC, ANCC, APS, MR, UR, HR] as defined in Section 2. If the weight $w(e_{ij})$ is greater than the threshold ϵ, then an edge connecting node *i* and node *j* is added to the graph with weight equals to $w(e_{ij})$ (see Figure 1, the thickness of an edge reflects its weight).

Afterwards, every node enlarges the similarity graph further by exploring the possibility of creating more similarity edges with the neighbors of its currently direct neighbors.

3.2 Clustering by Community Detection

Our objective is to find the topological communities inside the constructed similarity graph. Let us first define similarity graph as an undirected weighted graph G = (V, E), where V is the set of nodes (or users) and E is the set of similarity edges, where $e_{ij} \in E$ denotes cosine similarity between nodes v_i and $v_j \in V$ that is computed as defined in Equation 5. Commonly, finding communities is well-know as community detection and is defined as:

Definition 3.1. ***Community Detection****. A community detection C, also known as graph clustering, is a mapping*

$$C : G \rightarrow G'_1 \times ... \times G'_n \tag{6}$$

that partitions G into n non-empty, node-disjoint subgraphs $G'_1 \times ... \times G'_n$ representing a set of communities or clusters. A widely used quality measure for community detection is the modularity Q of the clustering C(G) [16], which is a mapping

$$Q : C(G) \rightarrow \mathbb{R} \tag{7}$$

that assigns a quality value q to the clustering C(G), as defined by

$$q := \sum_i (w(e_{ii}) - b_i^2) \tag{8}$$

Where $b_i = \sum_j w(e_{ij})$, where e_{ij} represents an edge in community i for which the target node of the edge lies in community j. The higher the quality value q is, the better the detected community is. One possible definition for C is to maximize Q over all clustering C(G) [16].

Algorithm 1: Community Detection Methods

Result: Community Structure C_{t+1}
Procedure `selectCommunity` (*node* u)
  forall the $C \in NeighborCommunity(u)$ **do**
    $q(C) \leftarrow sum(w_{e_{uj}}) | C_j = C$
  end
  $C_u \leftarrow C_j | q(C_j) = max(q(C))$
Procedure `changeCommunity` (*node* u)
  $C_{u_{new}} \leftarrow$ selectCommunity(u)
  if $C_u \neq C_{u_{new}}$ **then**
    $C_u \leftarrow C_{u_{new}}$
    forall the $x \in Neighbor(u)$ **do**
      changeCommunity(x)
    end
  end

Our appraoch employs recently developed decentralized diffusion-based community detection strategy [18]. In particular, every node in the similarity graph starts by joining the node with the maximum cosine similarity among its direct friends to form a community. Afterwards, in successive iterations every node chooses to quit its current community and join one of its neighbour's if this brings some modularity gains. As described in method *selectCommunity* in Algorithm 1, nodes select the dominant community in their neighborhood to join (i.e., the community with the highest sum of weights). This step is iteratively repeated until no node wants to change its community as it already represents the dominant one of all its neighbors. Thereafter, the topological communities detected in the similarity graph represent the different behavioral patterns associated with the direct friends of a user in the social graph.

3.3 Community Structure Adaptation

OSNs are dynamic by nature due to rapidly evolving social activities and interactions among users. Therefore, the constructed similarity graph must be continuously updated to cope with evolving users' behaviors. Thus, we have integrated adaptive modularity-based methods for identifying and tracing the changes in the communities structure of

the constructed similarity graph. The similarity graph is updated by either inserting or removing a node or set of nodes, or by either introducing or deleting an edge or set of edges. We have modeled these graph changes as a collection of simple events namely: *newNode*, *removeNode*, *newEdge* and *removeEdge* whose details are as follow:

- newNode *(V+u)*: a new node *u* with its associated edges are introduced, such that *u* could come with no or more than one new edge(s).
- removeNode *(V-u)*: a node *u* with its adjacent edges are removed from the graph.
- newEdge *(E+e)*: a new edge *e* connecting two existing nodes is introduced.
- removeEdge *(E-e)*: an existing edge *e* in the graph is removed.

Algorithm 2: Node Simple Events

Result: An updated Community Structure C_{t+1}
Procedure `newNode` (*node* u)
  $C_u \leftarrow$ selectCommunity(u)
  Update C_{t+1} : $C_{t+1} \leftarrow (C_t \setminus C_u) \cup (C_u \cup u)$
Procedure `removeNode` (*node* u)
  $C_u \leftarrow (C_u \setminus u)$
  forall the $v \in Neighbor(u)$ **do**
    removeEdge(e_{vu})
  end

Our approach starts by extracting initial community structure C_0, by detecting the communities exist in the first snapshot of the network. Afterwards, this initial structure is continuously updated for the successive snapshots by applying the above methods as illustrated in Algorithm 2 and Algorithm 3.

3.3.1 New node

When a new node joins the similarity graph, it assigns itself to the dominant community in its neighborhood as illustrated in method *newNode*.

3.3.2 New edge

In case of a new edge *e* is introduced, we can divide it further into two cases: an intra-community edge (both nodes belong to the same community) or an inter-community edge (connecting two communities). In the first case, no change happens to the community structure (as detailed in method *newEdge*). Yet, the interesting situation happens when *e* is an inter-community edge, as its presence could possibly make source and destination nodes change their community memberships. Consequently, these nodes notify their neighbors in case of change, so as cascading updates could take place if further changes are required (as detailed in method *changeCommunity*).

3.3.3 Remove node

As shown in methods *removeNode*, when an existing node *u* is of a community *C* is removed, all of its adjacent edges are removed as a result. Consequently, the resulting community structure might change, hence, neighbors of that removed node re-evaluate their community memberships as illustrated in the next method *removeEdge*.

Algorithm 3: Edge Simple Events

Result: An updated Community Structure C_{t+1}
Procedure `newEdge` (*edge* e_{vu})
  if v *and* u *are new nodes* **then**
    $C_{t+1} \leftarrow C_t \cup \{v, u\}$
  else if $C_v = C_u$ **then**
    $C_{t+1} \leftarrow C_t$
  else
    changeCommunity(u)
    changeCommunity(v)
Procedure `removeEdge` (*edge* e_{vu})
  if (v, u) *is a single edge* **then**
    $C_{t+1} \leftarrow (C_t \setminus \{v, u\}) \cup \{v\} \cup \{u\}$
  else if *either* v *(or* u*) is of degree one* **then**
    $C_{t+1} \leftarrow (C_t \setminus C_v) \cup \{v\} \cup \{C_v \setminus v\}$
  else if $C_v \neq C_u$ **then**
    $C_{t+1} \leftarrow C_t$
  else
    changeCommunity(u)
    changeCommunity(v)

3.3.4 Remove edge

Similarly to edge addition, edge removal can be divided into two case, such that the edge to be removed *e* is either an inter-community edge or intra-community edge. In the first case, the removal of *e* will strengthen the current community structure and cause no change to it. However, in the second case, edge removal might cause community split. Therefore, the edge source and destination nodes re-evaluate their community memberships and notify their neighbors in case of change.

3.4 Lexical Analysis of Posts

As aforementioned, the core of our approach is to detect different behavioral patterns in the user similarity graph by performing community detection. However, spotting spam patters among detected ones is not straightforward. So as, further lexical and semantic analysis is required to efficiently spot spammers communities among extracted ones. Specially, spammers can use automated spinning to avoid duplicate detection, such that they can create new versions with vaguely similar meaning but sufficiently different appearance. Therefore, in our approach we apply lexical analysis of the most frequent words to determine if those words or their synonyms are commonly used by spammers. Our approach integrates Gavagai lexicon[2] that learns the words synonyms and their related n-grams terms.

Many vector-based language models have been used for the purpose of representing word meanings. There models represent each word type by a single vector of contextual features obtained from co-occurrence counts in large textual corpora. Random Indexing [11], that is implemented in Gavagai lexicon, uses sparse and high-dimensional vectors to represent words. Interestingly, random index representation depends on the dynamic usage of a term in a corpora. In particular, fixed-size words vectors are initialized

[2]Available via http://lexicon.gavagai.se/

randomly, then words vectors are continuously updated with every document in the corpora by means of context windows. So as, word vectors are accumulated by vectors of a few adjacent words appearing on each side of the focus word in each parsed document in a corpora.

Accordingly, in our approach we identify a set of trusted nodes in the social graph and these nodes are responsible for labeling any of detected communities as spam if the majority of the users belonging to these communities frequently use spam words or their lexical related terms.

3.5 Complexity Analysis

The model's cost is expected to be low given that every node performs its local computation independently of the other nodes. We discuss the complexity of our approach in terms of communication traffic between all the nodes in the OSN. By our adopted work for decentralized community detection, the algorithm's complexity is a $\mathcal{O}(N * D * R)$, where N is the total number of users in the similarity graph, D is the average node degree, and R is the total number of rounds needed for the algorithm to converge[3] [18]. This step requires that all the nodes are online at the time of its execution; however, it is also a process that is performed once and that is incrementally updated only. Moreover, as we demonstrate through experiments on real OSN data, the convergence time of our solution is very realistic and achievable (see Section 4.5).

4 Evaluation

Our approach applies vertex-centric approach which is proved to be scalable, efficient and fast. Our algorithms are implemented in GraphLab[4], with two different distributed execution modules. In the first module, nodes participate in creating the similarity graph using their feature vectors. Thereafter, the control is moved to the second module that performs the community detection algorithm. In the following subsections, we thoroughly evaluate the performance of our approach in terms of the accuracy of spam detection. We compare our method with different centralized and supervised binary classification approaches, utilizing the Weka tool[5], namely: K-means (KM) with number of clusters =2, Decision Tree (DT) and Random Forest (RF).

[3] R depends on the topological properties of the underlying graph
[4] https://turi.com/products/create/
[5] http://www.cs.waikato.ac.nz/ml/weka/

Table 1: Twitter datasets used in our experiments.

Twitter Dataset	(1)	(2)	(3)
Tweets	453,519	489,484	360,927
Legitimate Accounts	17,322	19,312	12,128
Suspended Accounts	2,072	1,617	3,109
Social-graph Edges	1,357,806	1,187,036	2,349,314
Similarity-graph Edges	2,149,414	2,297,150	3,339,617

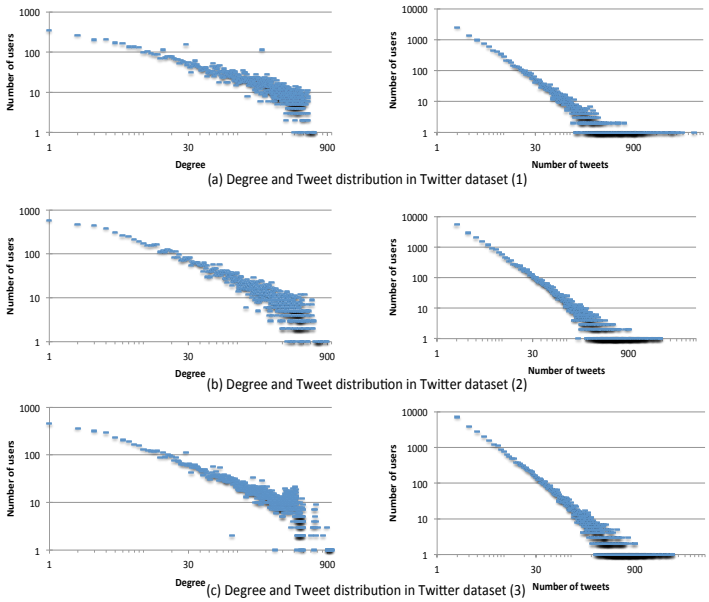


Figure 2: The degree and tweet distributions for the collected datasets in log scale.

4.1 Datasets

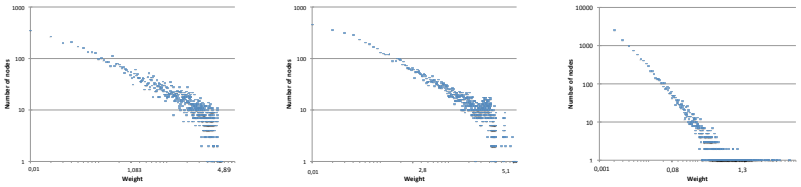
We collected our dataset from Twitter using Twitter streaming API[6] during fourteen month period from May 2015 to July 2016. We access Twitter's API using privileged accounts, collecting users' tweets and the social graph connecting these users. Particularly, we have started by generating random user identifiers for each dataset, then collect social graph surrounding these initial nodes. Iwc n order to identify the spammers within our datasets, we query the status of all accounts regularly to check if any got suspended for abusive behavior. Upon suspension, we identify suspended accounts as spammers. Table 1 lists the details of the collected datasets.

Additionally, in Figure 2 we show the degree and tweets distributions for the collected datasets in log scale. As shown, both of degree and tweet distributions follow power law probability distribution, such that there is uneven distribution of connections and tweeting behavior. Some nodes have very high degrees of connectivity (i.e., hubs), while most have small degrees. Similarly, majority of users post few tweets, whereas there is small number of highly active users who post large number of tweets.

4.2 Generated Similarity Graph

As aforementioned, the user similarity graph is constructed in a fully decentralized manner, such that each node explores its surrounding neighborhood progressively to add further similarity edges. Furthermore, as mentioned in Section 3.1, nodes add similarity edges if the similarity weight is greater than the threshold ϵ. We allow nodes to determine freely the value of ϵ, such that each nodes computes the average weight of its current edges, and set ϵ's value to that average. As shown in Table 1, the average number of added edges in the similarity graph is almost equals to 50% of the existing edges in the

[6]https://dev.twitter.com/rest/public



(a) Twitter dataset (1) (b) Twitter dataset (2) (c) Twitter dataset (3)

Figure 3: The Similarity weight distributions for the collected datasets in log scale.

social graph. According, our approach connects only highly similar nodes instead of creating a fully connected graphs that can be difficult to be partitioned correctly.

Figure 3 depicts the similarity weight distribution obtained for each dataset in log scale. As shown, the similarity weight distribution follows power law probability distribution similarly to the degree and tweet distributions. Furthermore, the similarity weight distribution spans over wider range in datasets (1) and (2) compared to dataset (3). Particularly, in dataset (3) 91.5% of the similarity weight is less than 0.25, and this resulted from the low post frequency that users in this dataset have as shown in Figure 2. Specifically, the users in both if these two datasets (1 and 2) have similar tweeting behavior in terms of distribution, such that the frequency of users with few posts is low (between 2.5k and 5k) compared to the situation in last dataset (around 7k). Therefore, we can infer that the more active posting behavior of users, the more strong edges will be added in the similarity graph. Additionally, our approach can successfully adapt to different social activities of the users and accordingly create the user similarity graph to reflect the underlying user behavior.

4.3 Extracted Communities

As aforementioned, every node repeatedly runs the community detection, until communities structure does not change any more (i.e., the convergence is reached). Figure 4 depicts the number of rounds required till convergence, and number of extracted communities per round. As shown, in the very beginning the number of communities is very large, every node starts to form a community with one of its direct neighbors. However, over time nodes join the dominant communities in their neighborhood, as a result the communities start to merge and the number of communities continues to decrease.

In order to identify the communities the contain spammers, we have construct a list of 500 words that are commonly used by spammers associated with their semantically similar terms and n-grams (see Section 3.4). Further, for every node we select the most frequent words used in its tweets. Accordingly, the collected word list per community is checked against a list common spam words. A community is identified as spam if majority (i.e., more than 50%) of its members use common spam words in their tweets. The results show that the percentage of spam communities is 17.3%, 21.6% and 23.5% in datasets (1), (2) and (3), respectively.

4.4 Performance Comparison

We calculate the accuracy of our approach using True Positive Rate and False Positive Rate, that are defined as the following:

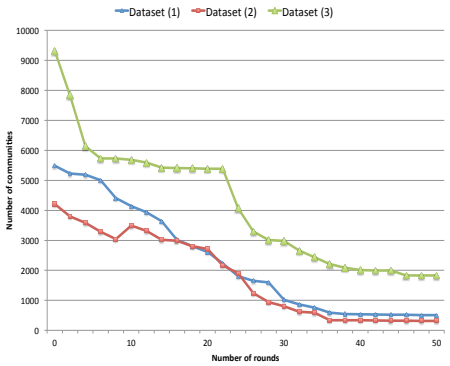
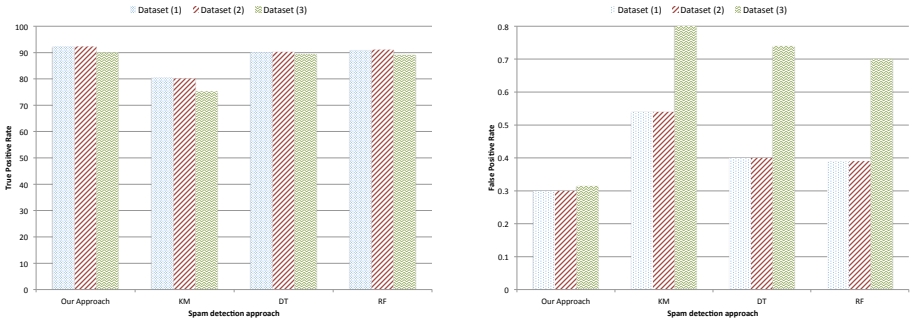


Figure 4: The number of rounds required for convergence.



(a) True positive rate

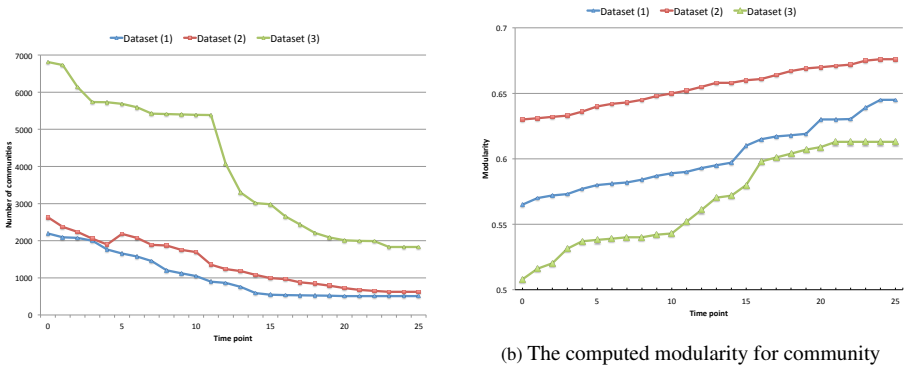
(b) False positive rate

Figure 5: The performance achieved by our approach compared with supervised methods.

- *True Positive Rate (TPR)*: we calculate TPR as the fraction of spammers that are successfully detected.
- *False Positive Rate (FPR)*: we calculate FPR as the fraction of legitimate users that are identified as spammers.

Figure 5 depicts the detection performance comparison of our approach with the different centralized and supervised classification methods. As shown, our approach outperforms all binary classification methods. Furthermore, though our approach is a decentralized one, yet our performance is slightly better than other centralized methods. Specifically, we can find that the TPR of our approach is the highest (92.3%). Therefore, the reported result of TPR confirms the ability of graph-based clustering to achieve more accurate detection rate compared to the binary classification.

Additionally, our approach has the lowest FPR, which means that graph-based clustering successfully detect spammers with minimum affect on the legitimate users. Specifically, we can see that FPR in our approach can be steadily maintained under 0.3%, as shown in Figure 5, while this rate is 0.39% for the best binary classification method (RF). Consequently, the community detection approach adopted in our approach perfectly categorizes the existing behavioral patterns into more homogeneous and accurate clusters than binary classification.



(a) The number of detected communities

(b) The computed modularity for community detection

Figure 6: The adaotation of our approach with evolving graphs.

4.5 Community Adaptation

In this experiment we study the adaptability of our approach with dynamic and evolving graphs. We started by loading 50% to form the basic structure, such that we constructed the similarity graph using only 50% of nodes from the social graph and 50% of their associated tweets. Afterwards, we simulated the network evolution by adding the remaining nodes/tweets via a series of 25 growing snapshots. Figure 6 depicts the number of detected communities per snapshotas well as the resulted modularity of the detected communities per snapshot.

Additionally, we have noticed that the changes caused by incrementally adding the snapshots are localized, such that on average 15% to 17% of old nodes got affected by the change and re-evaluate their communities membership. Consequently, our approach dynamically adapt to the topological changes of evolving graphs. Moreover, our approach adapts incrementally with no need to start community detection from scratch.

5 Related Work

The first family of spam detection mechanisms includes techniques using blacklists to identify URL on OSNs websites directing to spam content [8, 10, 20]. However, URL blacklisting has several practical challenges. First, those blacklists are publicly available, hence spammers can evade them by changing their domain names or hiding them behind some redirecting pages. Second, URL blacklisting becomes ineffective with the spread usage of URL shortening services such as bit.ly and t.co. For example, spammers can generate many shortened URLs for their pages without being listed. Therefore, to overcome the wide-spread abuse shortened URLs, different techniques have been proposed to analyze the redirection chains of URLs and their correlations [12, 20, 13]. Yet, those techniques are not designed as online detection tools, since they either have long lag-time or limited efficiency.

Furthermore, a rich corpus of research work lies in adopting supervised machine learning based methods using hybrid features extracted from textual content and OSN friendship graph. For example, Hongyu et al. [7] propose to train a binary classifier with user-based features that distinguish spam from legitimate content. The user social degree is used among, yet spammers can increase their social degree by purchasing

more followers. Yang et al. [21] propose using graph-based features that are hard to fake such as local clustering coefficient and betweenness centrality. The main objective of spam is reaching a wide audience, so as spammers start to hijack trending topics and include many accessible accounts by intentionally mentioning them in their spam posts. Therefore, Amleshwaram et al. [1] define new content features that measure how spammers entrap victims by counting the number of unique mentions and hijacked trending topics embedded in the posts.

More recently, [19] suggests an unsupervised solution to spam detection based on sybil defense mechanism. The proposed scheme starts by identifying non-spammers (i.e., non-sybils) by applying a clustering algorithm on social graph. Additionally, the authors focus their analysis on intensive URL sharing, yet instead of using URL blacklisting, they add new user-link edges to the social graph by connecting users sharing the same URL. Afterwards, they filter the graph by removing non-sybil nodes and nodes with low degree, then the remaining nodes are identified as spammers. However, the assumption that sybil nodes form tight-knit communities does not presist as shown in recent studies [24].

6 Conclusion

In this paper, we have introduced a novel decentralized and unsupervised spam detection framework in contrast to existing centralized and supervised approaches. Our approach resembles graph-based spam detection technique that detects spam using graph clustering on top of a newly constructed user similarity graph which encodes within its topology a holistic view of all behavioral interactions and patterns of OSN users. Particularly, our approach employs both content-based and graph-based features to encode user behavioral similarity. More importantly, our approach integrates community detection algorithm that categorizes the existing user behavioral patterns into more homogeneous and accurate clusters than binary classification. The proposed approach achieves detection accuracy upto 92.3% and false positive rate less than 0.3%. Additionally, our approach is scalable and massively parallel that suitably fits DOSNs and OSNs environments.

As a natural continuation of the work, we plan to increase the strength of Jaccard similarity by integrating semantic similarity measures to prevent content spinning. Specifically, spammers can use automated spinning to avoid duplicate detection, such that they can create new versions with vaguely similar meaning but sufficiently different appearance. Yet, combining Jaccard similarity with semantic and textual analytics measures can successfully spot spun content. Consequently, we will be able to spot spammers who share different variations of the similar-meaning content.

References

[1] Amit A Amleshwaram, Nutan Reddy, Suneel Yadav, Guofei Gu, and Chao Yang. Cats: Characterizing automation of twitter spammers. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–10. IEEE, 2013.

[2] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on twitter. In *CEAS'10*, volume 6, page 12, 2010.

[3] Chao Chen, Jun Zhang, Yang Xiang, Wanlei Zhou, and Jonathan Oliver. Spammers are becoming" smarter" on twitter. *IT Professional*, 18(2):66–70, 2016.

[4] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Handbook of Social Network Technologies and Applications*, pages 349–378. Springer, 2010.

[5] Sergey N Dorogovtsev and Jose FF Mendes. Evolution of networks. *Advances in physics*, 51(4):1079–1187, 2002.

[6] Marcel Flores and Aleksandar Kuzmanovic. Searching for spam: detecting fraudulent accounts via web search. In *Passive and Active Measurement*, pages 208–217. Springer, 2013.

[7] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. Towards online spam filtering in social networks. In *NDSS*, 2012.

[8] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. Detecting and characterizing social spam campaigns. In *SIGCOMM'10*, pages 35–47. ACM, 2010.

[9] Hongyu Gao, Yi Yang, Kai Bu, Yan Chen, Doug Downey, Kathy Lee, and Alok Choudhary. Spam ain't as diverse as it seems: throttling osn spam with templates underneath. In *ACSAC'14*, pages 76–85. ACM, 2014.

[10] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @ spam: the underground on 140 characters or less. In *CCS'10*, pages 27–37. ACM, 2010.

[11] Jussi Karlgren and Magnus Sahlgren. From words to understanding. In Hideki Asoh Yoshinori Uesaka, Pentti Kanerva, editor, *Foundations of Real-World Intelligence*, chapter 26, pages 294–308. CSLI Publications, 2001.

[12] Sangho Lee and Jong Kim. Warningbird: Detecting suspicious urls in twitter stream. In *NDSS*, 2012.

[13] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, et al. Click trajectories: End-to-end analysis of the spam value chain. In *SP Symposium*, pages 431–446. IEEE, 2011.

[14] Mary Madden, Amanda Lenhart, Sandra Cortesi, Urs Gasser, Maeve Duggan, Aaron Smith, and Meredith Beaton. Teens, social media, and privacy. *Pew Research Center*, 21, 2013.

[15] Juan Martinez-Romo and Lourdes Araujo. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications*, 40(8):2992–3000, 2013.

[16] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[17] NEXGATE. 2013 State of Social Media Spam. `http://nexgate.com/solutions/nexgate-social-media-security-stat-center/`, 2013. [Online; accessed 08-August-2016].

[18] Fatemeh Rahimian, Sarunas Girdzijauskas, and Seif Haridi. Parallel community detection for cross-document coreference. In *IEEE/WIC/ACM*, volume 2, pages 46–53. IEEE, 2014.

[19] Enhua Tan, Lei Guo, Songqing Chen, Xiaodong Zhang, and Yihong Zhao. Unik: Unsupervised social network spam detection. In *CIKM'13*, pages 479–488. ACM, 2013.

[20] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *SP Symposium*, pages 447–462. IEEE, 2011.

[21] Chao Yang, Robert Harkreader, and Guofei Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *Information Forensics and Security*, 8(8):1280–1293, 2013.

[22] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *WWW'12*, pages 71–80. ACM, 2012.

[23] Chao Yang, Robert Chandler Harkreader, and Guofei Gu. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *Recent Advances in Intrusion Detection*, pages 318–337. Springer, 2011.

[24] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *TKDD'14*, 8(1):2, 2014.

[25] Yin Zhu, Xiao Wang, Erheng Zhong, Nathan Nan Liu, He Li, and Qiang Yang. Discovering spammers in social networks. In *AAAI'12*, 2012.