POISED: Spotting Twitter Spam Off the Beaten Paths

Shirin Nilizadeh
UC Santa Barbara
California, USA
nilizadeh@ucsb.edu

François Labrèche
Ecole Polytechnique de Montréal
Québec, Canada
francois.labreche@polymtl.ca

Alireza Sedighian
Ecole Polytechnique de Montréal
Québec, Canada
alireza.sadighian@polymtl.ca

Ali Zand
UC Santa Barbara
California, USA
zand@ucsb.edu

José Fernandez
Ecole Polytechnique de Montréal
Québec, Canada
jose.fernandez@polymtl.ca

Christopher Kruegel
UC Santa Barbara
California, USA
chris@ucsb.edu

Gianluca Stringhini
University College London
London, United Kingdom
g.stringhini@ucl.ac.uk

Giovanni Vigna
UC Santa Barbara
California, USA
vigna@ucsb.edu

ABSTRACT

Cybercriminals have found in online social networks a propitious medium to spread spam and malicious content. Existing techniques for detecting spam include predicting the trustworthiness of accounts and analyzing the content of these messages. However, advanced attackers can still successfully evade these defenses.

Online social networks bring people who have personal connections or share common interests to form communities. In this paper, we first show that users within a networked community share some topics of interest. Moreover, content shared on these social network tend to propagate according to the interests of people. Dissemination paths may emerge where some communities post similar messages, based on the interests of those communities. Spam and other malicious content, on the other hand, follow different spreading patterns.

In this paper, we follow this insight and present POISED, a system that leverages the differences in propagation between benign and malicious messages on social networks to identify spam and other unwanted content. We test our system on a dataset of 1.3M tweets collected from 64K users, and we show that our approach is effective in detecting malicious messages, reaching 91% precision and 93% recall. We also show that POISED's detection is more comprehensive than previous systems, by comparing it to three state-of-the-art spam detection systems that have been proposed by the research community in the past. POISED significantly outperforms each of these systems. Moreover, through simulations, we show how POISED is effective in the early detection of spam messages and how it is resilient against two well-known adversarial machine learning attacks.

CCS CONCEPTS

• **Security and privacy → Social network security and privacy**; *Intrusion/anomaly detection and malware mitigation*;

KEYWORDS

Spam Detection; Online Social Networks; Information Diffusion; Communities of Interest; Parties of Interest

1 INTRODUCTION

Cybercriminals have found in social networks a propitious medium to spread malicious content and perform scams against users [36]. Social networks are leveraged by cybercriminals for a number of reasons. First, social networks are very popular, with the largest ones having hundreds of millions of users: this constitutes a large victim base for criminals. Second, attackers who compromise social network accounts with an already established reputation can exploit the inherent trust between connected users to spread malicious content very effectively [27, 38].

Previous work addressed the detection of spam on social networks by predicting the trustworthiness of the accounts that post messages, notably by detecting Sybil communities [22, 87], bots [29, 75, 86], compromised accounts [27], or a combination of these [13, 85]. Recent research, however, showed how attackers can successfully evade both Sybil-based defenses [50] and account-based ones [93]. This happens because existing spam detection systems detect *the way in which malicious accounts infiltrate the network and build connections*, rather than *the way in which malicious messages spread across the network in comparison to legitimate ones*.

In this paper, we propose a novel way to detect malicious messages on social networks. Instead of looking at the characteristics of accounts or messages, we inspect the way in which messages spread on the social network.

In social networks, users tend to form *networked communities*, where most users are connected to many other users within the same community. These communities can be recognized by their structure in the underlying connection graph, as they form strongly connected subgraphs. For this reason, they are also dubbed *structural communities*. The reasons why such communities form are as varied as the reasons why people connect to each other, such as family, geographical location, past common history, etc. Nonetheless, it has been recognized that one important reason why members of social networks tend to connect to others is the so-called *homophily principle*, *i.e.,* people connect to other people who hold similar thoughts and values [54]. In that sense, these people also form *communities of interest*, where connected users communicate and interact on topics of common interest. In principle, members of

a networked community may not necessarily share the same interests, and thus, structural communities and communities of interest may not coincide. However, we postulate that the homophily principle constitutes indeed the principal reason why people connect. Therefore, we formalize our first hypothesis as follows:

H1: In social networks, the topics of interest of users within a networked community are strongly shared among its members. In other words, networked communities are structured subsets of the larger set of users interested in the same topics.

It is recognized that the topology of social networks has an extremely important role in the dissemination of information [59, 90]. The dissemination of information is shaped by the structure of the network, and in particular faster dissemination is favored within networked communities [48]. Nonetheless, such dissemination can traverse networked communities as long as there are members in both communities who share the same interests. As time progresses, dissemination paths may emerge where some communities trigger and post specific messages based on the interests of those communities and of the surrounding ones. These dissemination paths can help us predict patterns of postings within and outside of communities. For example, if two communities C_1 and C_2 always post messages on similar topics, then when a message is observed in C_1, the same or similar message has a high probability to also be posted or shared in C_2. In this paper, we refer to these communities interested in the same set of topics as *parties of interest*.

On the other hand, spam typically spread differently throughout the network. For example, messages that are posted by compromised accounts may spread in unexpected communities, because each compromised user posts that message regardless of whether the topic is of interest to the account owner or of the communities of interest of which the compromised user is a member [28]. This leads to formulate our second hypothesis as follows:

H2: Normal messages disseminate through predictable *parties of interest* that include intra-community communication and inter-community exchanges between structural communities that share common interests. Conversely, the propagation probability of malicious messages through these parties of interest do not match with those of normal messages.

In this paper, we investigate these two hypotheses through experimentation on the Twitter social network. First, our analysis shows that, on Twitter, community members have a similar and restricted set of topics of interest, thus validating our first hypothesis. We then build a system, called POISED, that is able to detect whether a message shared on a social network spreads through expected parties of interest, or if it rather spreads anomalously. Our experimental performance evaluation shows that POISED can detect malicious spam messages with high accuracy, thus validating our second hypothesis.

In a nutshell, POISED works as follows. First, it detects networked communities in Twitter by partitioning its social graph. Second, it identifies topics of interest in these communities. Then, it tracks the dissemination of similar messages through communities and constructs a probabilistic model of the parties of interest through which these messages are normally disseminated. Finally, leveraging this model, a classifier detects malicious content by identifying the messages that do not follow these expected parties of interest. POISED can successfully detect spam messages with 91% precision and 93% recall. We also compare POISED to three state-of-the-art spam detection systems that have been proposed by previous work: SpamDetector [75], Compa [27], and BotOrNot [25]. With respect to the F1-score, POISED outperformed them by more than 70%, 35%, and 83%, accordingly.

Through simulations, we show that POISED performs very well in detecting spam messages early on. For example, it can detect spam messages that have spread through only 20% of the communities with 88% precision and 75% recall.

Finally, we investigate the resilience of POISED against two common adversarial machine learning attacks [46], *poisoning* [70] and *evasion* attacks [5]. Our simulation results suggest that the adversary needs to have a great knowledge about the network and parties of interest to highly impact the performance of POISED. For example, even if 30% of the network is compromised, the precision and recall remain at 82% and 87%, in the case of a poisoning attack, and at 75% and 52%, in the case of an evasion attack.

In summary, this paper makes the following contributions:

(1) Through our experiments on 300 Twitter neighborhoods with more than 15M tweets and 82K users, we show that networked communities are built around shared topics.
(2) We developed POISED, which relies on a combination of techniques from network science, natural language processing, and machine learning to detect spam messages by predicting the dissemination of messages through parties of interest. We tested POISED on a ground-truth dataset including data for 202 neighborhoods in Twitter with about 1.3M tweets and 64K users. Our results suggest that our approach is successful in detecting spam messages. Moreover, it outperforms other state-of-the-art detection systems.
(3) Our results demonstrate that this approach is scalable and can detect spam with only a partial knowledge of the social network. Our simulation results show that spam messages can be detected early on, when only attaining 20% of their potential reach in their neighborhood network. We also show that POISED is difficult to evade for an active adversary. We simulate two attacks in which the adversary attempts to mimic the propagation of benign messages, and show that POISED is still highly effective even when the adversary has compromised a large portion of the network.

2 BACKGROUND AND THREAT MODEL

2.1 Threat Model

In our threat model, spam messages are posted on a large scale [81, 83] and are similar in content and format since, in most cases, they are generated by similar templates [31, 32]. This can be accomplished either by creating fake (Sybil) accounts [22], compromising and abusing legitimate accounts [27], or by purchasing bots [30].

Unlike other related work that focuses on analyzing message content (*i.e.,* URLs) [47, 80], or finding compromised accounts [22, 28], we do not place any additional constraints on the type of spam messages sent nor on the type of accounts used by the adversary.

Spam detection is an adversarial problem. In a real setting, an adversary could reverse engineer how POISED works and actively attempt to evade it. Therefore, we assume that the adversary is able

to post spam messages through parties of interest similar to those of benign messages. Particularly, we assume that the community detection and topic detection algorithms can be played. Malicious accounts can compromise some parts of the network, establish connections with honest users and pretend to share the same interests as the target communities. We also assume they can replicate the propagation model of benign messages through the parties of interest. For example, an attacker may observe the number of times a specific benign (viral) message has been posted in compromised communities as well as the number of users who have posted those messages, and then generate or compromise accounts to imitate legitimate parties of interest.

2.2 Communities and Parties of Interest

In social networks, users establish connections with others, and through them are able to interact with each other. The structure of the underlying connection graph is not homogeneous: rather than being connected with any user in the network, users tend to connect to each other, creating networked communities, where "everybody knows everybody." On the other hand, users tend to come together and form groups to interact around specific topics of interest [20, 39, 49], *i.e.,* that they tend to form *communities of interest.* Thus, the question becomes whether networked communities, defined in terms of the actual connections in the network, coincide with this topic-oriented notion of communities of interest.

The homophily principle [54] has been observed on online social networks, where users tend to connect to people who hold similar thoughts and values [42, 89]. In summary, this research would seem to suggest that the concepts of community and topics of interest are related.

However, they cannot be exactly the same. Indeed, we expect that in the social network universe there may be several different groups of users that are interested in the same things, but that are not in direct contact with each other. In other words, there may exist several networked communities that share the same interest, but that are not connected. In that case, the overall community of interest would consist of a set of disconnected networked communities, and would thus not constitute a networked community *per se.* Thus, modulo this caveat, one can postulate that networked communities do constitute communities of interest, albeit not a *complete* community of interest regrouping *all* users interested in those topics. This is indeed our first hypothesis.

In fact, this hypothesis has been implicitly employed in previous work on communities of interest. Indeed, in modern Internet-based social networks it is much easier to determine who is connected to whom than it is to determine *a priori* what are the topics of interest of users, let alone of groups of users. Thus, a proxy method for reconstructing these communities of interest has been to extract networked communities from the information on connections between users [79]. This can be done by using graph algorithms to identify dense subgraphs within the graph of user nodes and connections. We explore the validity of this hypothesis on our Twitter dataset.

In Figure 1, we provide an example of three such Twitter networked communities. The edges represent the "following" relationships between users. The size of nodes is scaled according to their

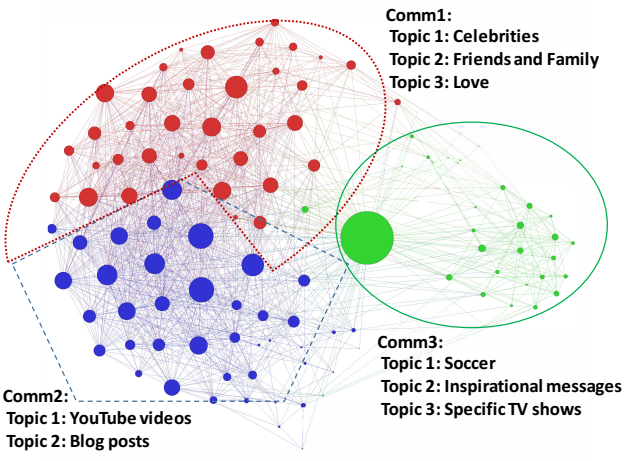


Figure 1: An example of three Twitter communities, where each is talking about a specific set of topics.

degree. We detected these three communities in a subgraph of 2000 users and extracted the topics of their members' discussions. We found that each community talks about a specific set of main topics. Users in *Comm1* post news about Hollywood celebrities and discuss topics about friends and family. Users in *Comm2* share videos of some specific reputable Youtube users and bloggers. Users in *Comm3* are interested in soccer, post inspirational quotes, and talk about a specific TV show.

Homophily [54] as well as the topology of social networks [59, 90] highly impact the dissemination of information in social networks. For example, by analyzing the tweets of communities, it is possible to predict viral memes [90]. On social networks, however, not all messages become viral [48] and many just travel through some networked communities interested in similar topics. The probability that a certain message is propagated through a set of networked communities is different from another set of communities in the network. We call these sets of communities, who care more or less about a message, the *Parties of Interest.* We hypothesize that the propagation probability of malicious messages throughout the network distinguishes them from the normal messages.

Based on this intuition, we propose a method for detecting malicious content on Twitter, called POISED (Parties Of Interest Semantic Extraction and Discovery). POISED computes and learns the propagation probability of messages in the networked communities and extracts the parties of interest. It then detects malicious messages by distinguishing their propagation probabilities and parties of interest from those of normal messages.

3 METHODOLOGY

The components of our system are shown in Figure 2: 1) data extraction, 2) community detection, 3) topic detection, 4) clustering of similar messages, and 5) spam detection. In the following, we explain each component in detail.

3.1 Data Extraction

We evaluated POISED on a large-scale dataset extracted from Twitter. Twitter is one of the most popular microblogging platforms with over 320 million active users [84]. This platform enables users to broadcast and share information. A user's timeline includes all *tweet* messages posted by that user. On Twitter, users follow others or are followed. Followers of a user receive all the tweet messages posted by this user. Twitter also provides a "retweet" mechanism that permits users to spread information of their choice beyond the reach of the original tweet's followers. Throughout this paper, we use the terms *messages*, *posts*, and *tweets* interchangeably. POISED utilizes user timelines and the social network. Here, we formally define a network as:

Definition 3.1. A *social graph* $G\langle V, E\rangle$ is a set of vertices V representing the users in the network and a set of edges $E \subseteq \{(u, v) : u, v \in V\}$ representing the set of social connections.

Note that a social graph G can be directed or undirected. If it is undirected, this means that $(u, v) \in E \iff (v, u) \in E$; If, on the contrary, it is directed, then $(u, v) \in E$ does not necessarily mean that $(v, u) \in E$, *i.e.,* u might be connected with v but not vice-versa.

3.2 Community Detection

Although there is no universally agreed-upon definition of a community in a social network, in a graph, structural communities usually refer to a group of nodes that are densely connected to each other and loosely connected to the rest of the graph. The nodes inside such a community might also share common properties and/or play similar roles within the graph. In social media, communities might have a link with external real entities. For example, a user might have a group of friends from the same city, a group from the same school, and yet another group interested in information security. Although these communities might be roughly defined and be overlapping with each other [1, 62], the concept behind them is still valid. In contrast, however, structural communities are normally defined as disjoint, non-overlapping sets of nodes of the graph. In this paper, we will favor the use of structural communities as they are easier to reconstruct from connection information. Later, our results suggest that detecting structural communities enables us to detect communities and parties of interest. Here, we define a networked community as follows:

Definition 3.2. A networked community structure C of a graph G is a disjoint partition of nodes in V, namely $C = \{C_1, \ldots, C_h\}$, where $C_i \subseteq V, V = C_1 \cup \cdots \cup C_h$, and $C_i \neq \emptyset, C_i \cap C_j = \emptyset$ if $i \neq j$, for all $i, j \in \{1, \ldots, h\}$. Nodes in a community C_i are connected to each other with higher probability than to nodes in other communities.

3.3 Topic Detection

Recently, natural language models have been used for clustering words in order to discover the underlying topics that are combined to form documents in a corpus. Topic detection algorithms such as Latent Dirichlet Allocation (LDA) [6] and Topic Mapping [45] have been successfully applied for analyzing text from user messages on social networks. By employing a topic detection algorithm, POISED identifies a set of topics of interest for a user and a community.

The LDA detection algorithm uses a list of documents as an input and detects the corresponding topics. For social networks with small message lengths, such as Twitter, topic detection is shown to be less efficient [37]. For this reason, we aggregate a user's messages into larger documents and then run topic modeling on the documents. For a user u, a set of documents D_u, namely $D_u = \{d_1, d_2, .., d_k\}$, is generated by partitioning the user's timeline into k groups with l messages. Note that l is the number of messages in a document and is a constant, whereas k varies based on the size of the user's timeline. Our evaluation with variation of $l = \{1, 5, 10, 20, 50, all\}$ shows that the length of documents do not have a significant impact on the overall results and therefore, we chose $l = 20$.

Formally, a user u's *topics of interest* (T_u), namely a list $T = \{t_1, t_2, \ldots, t_k\}$, consists of topics detected by a topic detection algorithm on user u's set of documents D_u.

Having extracted the topics for each user's documents, a community's topics of interest can be simply defined as the list of all topics detected for members of that community:

Definition 3.3. For a community C, its set of documents D_C is the union of the documents generated for each community member $u \in C$, $D_C = \bigcup_{u \in C} D_u = \{d_1, d_2, .., d_m\}$, where $m = \sum_{u \in C} |D_u|$.

Thus, the set of topics of interest for a community C is defined as follows:

Definition 3.4. The topics of interest T_C of a community C is a list $T_C = \{t_1, t_2, \ldots, t_m\}$ of m topics detected by a topic detection algorithm on the community's set of documents D_C, where each document d_i is labeled with a topic t_i.

Note that the topics in the topic lists T_u and T_C of a user and a community are not unique, and that several documents for the same user or the same community can be labeled with the same topic. Given Hypothesis 1, we expect that the topics found for a user or a community will greatly overlap.

Detecting topics of interest for each community, POISED constructs a network of structural communities where each community is represented by a set of topics.

3.4 Clustering Similar Messages

Similarity between messages can be measured by several metrics, and some can be more complex than others [23, 67, 82]. Other spam detection methods [27, 74] have effectively used an approach called *four-gram analysis* to identify similar messages on Twitter. This technique proved effective in our case : after manually inspecting the performance of this method on 60 clusters of different sizes, all clusters included the tweets with the same text, *i.e.,* all tweets were correctly grouped. As part of this approach, messages that share four or more consecutive words are clustered together. While in POISED, other algorithms can also be used to identify similar messages, in our evaluation we cluster messages employing four-gram analysis.

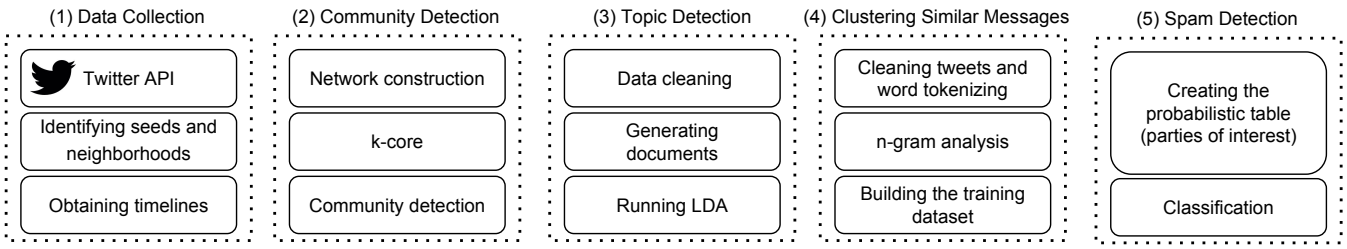


Figure 2: POISED **constructs a probabilistic model based on the diffusion of messages throughout communities of interest. Then, it employs supervised machine learning to classify messages as spam or benign.**

If a message contains less than four words, then all its words in their consecutive order are compared with other messages. The result of running this algorithm is a list of groups of similar messages, $g = [msg_1, msg_2, ..., msg_j]$, where g includes j similar messages, $|g| = j$. The messages in a group can be generated by a single user or several users.

3.5 Parties of Interest

A specific message could have been (re-)posted in one or several communities. Tracking the propagation of the messages in structural communities identifies *parties of interest*. Over time, this tracking makes it possible to predict the probability that a message posted in a community is also posted/retweeted in another community. POISED tracks the diffusion of messages over communities, and computes a probabilistic model for every cluster of similar messages.

First, the union set of topics of interest is constructed, which includes all topics detected from all documents. Then, for each group of similar messages, POISED counts the number of times that messages in the group have been observed in a community with a specific topic. These counts for a group are normalized by the total number of topics identified for messages in that group. For example, assume three communities, C_1, C_2 and C_3 are detected in a network, whose topics of interest are $\{t_1, t_2\}$, $\{t_1, t_3\}$ and $\{t_1, t_4, t_5\}$, respectively. Assume, a group includes three similar messages posted by users u_1 and u_2 in C_1, and user u_3 in C_2. The probability distribution for the union of topics in these three communities, $(t_1, t_2, t_3, t_4, t_5)$, is $\left(\frac{3}{6}, \frac{2}{6}, \frac{1}{6}, 0, 0\right)$. All three messages in this group are posted in communities with t_1 as their topic of interest (C_1 and C_2), while only one of these messages is posted in communities with t_3 as their topic of interest (C_2). Therefore, the count distribution for $(t_1, t_2, t_3, t_4, t_5)$, is $(3, 2, 1, 0, 0)$. The distributions are normalized by being divided by the message counts of the union of topics, *i.e.,* six in this example, to compute probability distribution.

As a result, for each group of similar messages, a probabilistic model is computed, which shows the potential parties of interest for that group of messages. Assume T is the union of all the detected topics in the dataset, where $T = \{t_1, t_2, ..., t_k\}$. Each group g of similar messages is represented with a topic probability vector $prob_g = [p_1, p_2, ..., p_k]$, where $k = |T|$ and p_j is the likelihood that messages in this group favor communities interested in topic t_j; this probability distribution thus represents the *parties of interest* for messages in g. The overall table of probability distributions for all groups of messages $prob\_table = \{probs_{g_1}, probs_{g_2}, ..., probs_{g_h}\}$ thus represents the *parties of interest* for the social network given the observed messages, and it is the basis for the classification model. In other words, using this model, if a message is seen in a specific community, it is possible to predict the probability that this (or a similar message) are going to be observed in other communities.

Note that POISED does not need to learn about the topics of messages, but only about their propagation through communities of interest.

3.6 Classification

If spam messages travel through different parties of interest than those of benign messages, then a classifier can learn these patterns and detect spam messages. Hence, by having a ground-truth data set, a classifier can be trained where topics found in communities are features of that classifier, and the class is defined as a binary variable that takes as values: *spam* or *benign*.

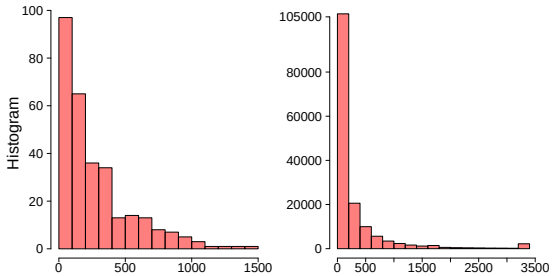
4 EVALUATION SETUP

4.1 Dataset

In December 2015, we used the Twitter API to crawl users' timelines. The API provides a stream of random users. We used a sample of 300 of them, called "seeds," and crawled the timelines for them as well as their friends and followers. Thus, we obtained data for 300 *neighborhoods* in Twitter, where a neighborhood consists of a seed with all his friends and followers. In our random selection of seeds, we did not collect data for users with more than 2,000 followers or friends, so that the crawling process would be of manageable size. We also limited the crawl to users having specified English as their language, so the further text analysis would be performed on tweets of a single language. To limit the bias of analysis in favor of older accounts with many tweets, and to work with more current data, we limited the number of tweets used per user to a maximum of 300 of their most recent English tweets. In our dataset, the average of all users' oldest and newest tweets are March 2015 and June 2015.

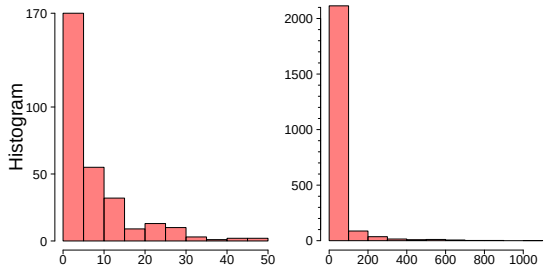
4.2 Network Construction

For each neighborhood, we constructed a directed and an undirected network based on *following* relationships between all users inside the neighborhood. The undirected network is obtained from the directed network, where the relationship between two users must



(a) Neighborhood size (b) Timeline length

Figure 3: While on average neighborhoods contain 270 users, some include more than 1000.



(a) Number of communities (b) Size of communities

Figure 4: Histograms for number of communities and community size.

be reciprocal to form an edge. In our experiments we examine the impact of both networks on our results.

Some users in each neighborhood only have a single connection to other users. These isolated users later result in some communities with a size of one and two. We applied a standard technique called k-core [72] to extract the maximal connected subgraph of each of the networks, where all nodes have a degree of at least k, here $k = 2$.

After obtaining the k-core of all 300 networks, the mean neighborhood size is 271, while some neighborhoods include more than 1000 users.

Figure 3(a) shows the histogram of the number of users (neighborhood size) in 300 neighborhoods. After obtaining the k-core of all 300 networks, the median for neighborhood size is 178 and the mean is 271. Seven neighborhoods include more than 1000 users. Figure 3(b) shows the histogram of timelines' length in our dataset, with an average value of 332 and a median of 177. Examining our hypotheses on both small and large neighborhoods demonstrates that our findings are not dependent on the size of a dataset or a specific neighborhood.

In summary, the dataset for testing the first hypothesis includes 15,751,198 English tweets posted by a total of 82,275 users in 300 neighborhoods.

4.3 Community Detection

To find structural communities, we employed one of the most widely accepted disjoint community detection algorithms, called Infomap [69]. This algorithm has shown good performance in tests using benchmark networks [43, 44].[1] In Infomap, a community is a partition that minimizes the average number of bits per step required to describe trajectories of random walkers.

Infomap detects a total of 2,283 communities in our 300 neighborhoods. While on average neighborhoods contain 8 communities, a handful of them contain more than 30 communities. Figure 4 shows the histograms for number of communities and their size resulting from the Infomap community detection algorithm. The median for number of communities in each neighborhood is 4 and the mean is about 7.5. The median for community size is 6 and the mean is about 30.

[1] Among the variety of community detection methods, we evaluated the impact of a set of them on our results, including Infomap [69], Spinglass [26], Walktrap [64], Leading eigenvector [60], Fastgreedy [18] and Multilevel [7]. Interestingly, we obtained very similar results while Infomap slightly outperforms the other algorithms.

4.4 Topic Detection

We applied Latent Dirichlet allocation (LDA) to find topics of interest for users and communities. We first generated text documents from users' timelines and then ran LDA modeling on the documents to obtain a list of detailed topics for each document.

Documents. As explained in Section 3.3, POISED divides the timeline into several documents.Since on Twitter, some accounts are older and some users post more often, the lengths of timelines can differ substantially. Figure 3(b) shows the histogram of timelines' length in our dataset, with an average value of 332 and a median of 177. On average, the users' timeline include 332 posts. As explained earlier, for each user we only consider at most 300 of the most recent tweets for our analysis.

Each document contains a fixed l number of tweets. The number of documents for a user depends on the number of tweets in the user's timeline. In our experiments, we investigated the impact of l on the list of detected topics and found that there is not a significant difference with a variation of l. Nonetheless, LDA detects topics slightly more accurately with 20 tweets per document.

Topics. We cleaned the documents by removing URLs, and non-printable characters. We removed *stop words*, a list of common words found in the English language, to improve topic detection and obtain detailed topics.

We employed the LDA implementation provided by Machine Learning for Language Toolkit (MALLET) [53]. The output includes documents labeled with a series of topics. We chose to label each document with the topic having the highest weight value. Having several documents for a user results in possibly several topics for the user. If all the documents of a user are labeled with a particular topic, then it shows that the user is interested in that specific topic.

MALLET requires a few parameters to apply LDA, such as the amount of topics to be found in the given documents. We experimented with various amounts of topics. Later, we show that the best results are obtained with 500 topics. Moreover, we tested different amounts of tweets per document, and later show that the best results are obtained with 20 tweets per document. In MALLET, we also set the iteration count to 200, which provides more precise topics at the expanse of a longer processing time.

5 EVALUATION: COMMUNITIES OF INTEREST

In this section, we examine our first hypothesis that members of a networked community have similar topics of interest.

5.1 Metrics and Null Model

We validate Hypothesis H1 on our dataset by computing three entropy-based metrics: *completeness*, *homogeneity*, and *V-measure* of topics detected in communities. These metrics were first proposed by Rosenberg and Hirschberg [68], and have been commonly used for evaluating many natural language processing tasks [19, 68].

All these three criteria produce a score in the interval of [0, 1], with 1 being 'good' and 0 'bad.' *Completeness* measures if all documents of a community are assigned to the same topic, *e.g.,* they are only about football. Symmetrically, *homogeneity* measures if each topic is only observed in a single community, *e.g.,* if all messages about a local art competition are posted by members of one community. *V-measure* is the harmonic mean of completeness and homogeneity and measures how successfully the two criteria are met. The computation of these measures is independent of the number of documents and topics in the communities and the network.

By computing these three metrics, we are able to measure if the members of a detected community are interested in the same topic.

To further validate that communities provide additional information about members' common interests, we propose comparing their scores with those of a *null model*. Statisticians use *null models* as baseline points of comparison for assessing goodness of fit [63]. Recently, null models have been used for studying network structures [16, 61, 63, 77] and community structures [56]. We generate a *null model* that randomly partitions documents into groups.

To have a fair comparison between the documents in the actual communities and those of the random clusters of the null model, the documents are shuffled so that the distribution of size of groups in the null model remains the same as in the detected communities in the network, and the number of communities remains the same. As a result, if the actual communities have higher scores, then our hypothesis is validated that users in communities are grouped over some specific topics of interest, and random clustering of documents does not provide similar or better scores.

5.2 Communities Discuss Different Matters While Community Members Talk About Similar Topics

We examine Hypothesis H1 by comparing the communities of interest that are detected in our neighborhoods with the random communities of the null model.

Figure 5 compares the scores for these two models. To examine the effect of the number of communities on the scores, Figure 5 presents them according to the number of communities in the neighborhoods. The *normal* results denote the scores obtained from the model created from actual communities, while the *random* results show the results for the null model created from random communities.

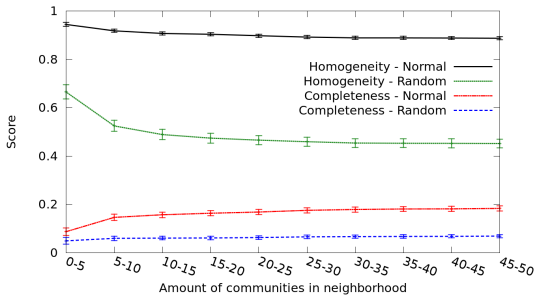


Figure 5: Both metrics highly decrease for the null model.

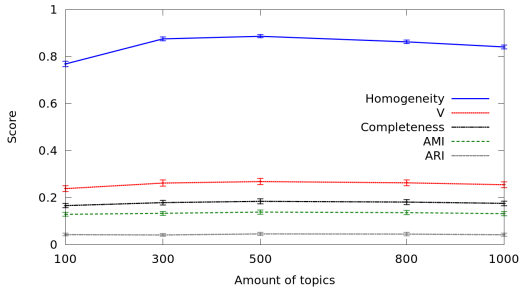


Figure 6: The metrics' score slightly increase for 500 topics.

Overall, the tests with random communities show a substantial drop in results. For communities of interest, on average, completeness and homogeneity scores are about 0.16 and 0.90 respectively while those scores are about 0.063 and 0.49 for random communities. We also ran a *Z-test* to compare the homogeneity and the completeness values for actual and random communities. For both metrics, the p-values are lower than 0.0001.

The completeness and homogeneity scores indicate that people in the communities talk about certain topics that are mainly different from those talked in other communities in their neighborhood. The homogeneity results also confirm that communities of interest are effectively distinguishable.

Some parameters can affect our evaluation of the first hypothesis. For example, the total amount of possible topics is a fixed value given as a parameter to MALLET. Also, we used a fixed amount of tweets written into each document. Figure 6 shows the impact of the amount of topics varying from 100 to 1000. As can be seen, the amount of topics do not significantly change the scores.

Only the homogeneity score slightly varied, being 0.87 at its maximum, when the topic count is 500. Similarly, we tested the impact of document size varying from 5 to 300 tweets per document. In summary, this parameter only affects the scores slightly (at most 0.17 in the range of [0,1]) and the best scores are provided when the document contains 20 tweets.

To measure the impact of different community detection algorithms, we also re-run the experiments for communities detected by several of these algorithms. The topic detection by LDA is independent of the community detection algorithm, and each community includes topics detected for its members.

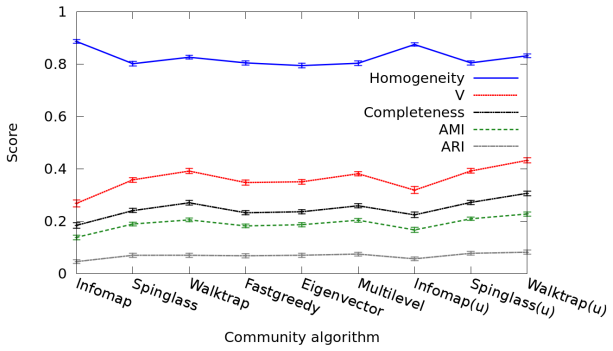
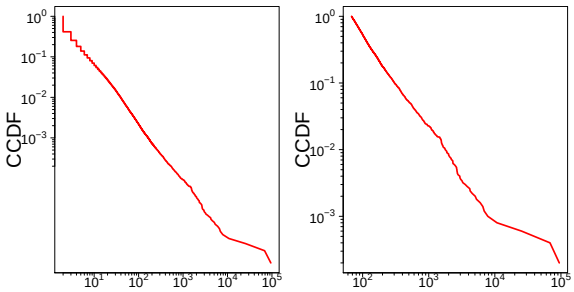


Figure 7: The communities and their topics of interest show high homogeneity but relatively low completeness.



(a) Global dataset (b) Labeled dataset

Figure 8: Size of groups of similar messages in the whole dataset and labeled dataset.

Figure 7 shows the scores for various community detection algorithms. These scores were calculated for each neighborhood, and then averaged. It illustrates that no matter what community detection algorithm is used, the communities and their topics of interest demonstrate high homogeneity, $[0.8, 0.89]$. Higher homogeneity confirms that communities do have little topics in common and, hence, are distinguishable. Considering its high homogeneity score, *Infomap* was chosen to be applied by POISED. Similarly, no matter what community detection algorithm is used, the completeness between communities and their topics is about 0.25. The *Walktrap* community detection algorithm provides communities with the highest completeness, with a score of 0.31. These values show that, for communities, not one but multiple topics are detectable.

Since the scores for the directed networks are slightly higher, we used the directed networks for our analysis.

6 EVALUATION: TWITTER SPAM DETECTION

Here, we examine our second hypothesis that benign and malicious content diffuse through distinguishable parties of interest, which can be used to detect Twitter spam messages.

6.1 Clustering Similar Messages

First, POISED needs to observe the diffusion of messages through communities of interest to learn about their parties of interest. For this, it applies four-gram analysis to detect similar messages in every neighborhood. We cleaned tweets by removing stop words and punctuation. Also, each URL is considered as a word.

In total, in our dataset, we found 1,219,991 groups of similar messages with the size range being between 2 and 94,382. Figure 8(a) shows the CCDF of the size of groups of similar messages in our dataset, which follows a power law distribution, with a small number of big groups and many small ones.

6.2 Create a Labeled Dataset

To evaluate the probabilistic models for spam detection, we need to have a ground-truth dataset including both spam and benign messages. Since it is not possible to manually label over 1 million clusters of similar messages, we picked the top 5000 groups after ordering them by their size, and labeled these groups. The group size picked for labeling ranges from 68 to 94,382. We may find more malicious campaigns by looking at larger clusters compared to all data. However, many large clusters also contain benign messages such as simple "happy birthday" wishes, as well as memes and trending topics.

The tweets in this dataset were manually checked by a group of 14 security researchers who labeled them independently, following a similar methodology to the one applied by previous work [14]. The researchers were advised on the risks of clicking on suspicious links and took precautions not to get infected (*e.g.*, by using virtual machines to lookup URLs). Each group of similar tweets is evaluated and labeled by three researchers, and then the majority vote is considered as the final label. We provided some guidelines for the researchers and defined some categories so that they can label each tweet with one of those categories. These categories are defined as:
Spam: A message that is encouraging users to do something, such as buying an item, voting for someone, visiting a URL, etc. Encouraging users is not by itself a malicious activity, however, if a tweet is doing it, then a more careful assessment is needed. For example, coders had to visit the websites these URLs refer to. If the URL is suspicious, then the message is spam. Note that memes also may include URLs. If in some cases, the URLs are not functional, then the coders were advised to label the groups based on their subject, and their tone.
App-generated: A message that is automatically posted by an application on the user timeline. Some examples are weather alerts posted by IFTTT[2] and health-related reports posted by fitness trackers such as Fitbit.[3] In the process, we also found that some of the bigger clusters are the result of some apps such as Twittascope, which regularly post tweets on behalf of the users. These tweets usually contain links to some articles.
Quote: A message that is a popular quote. In the first round of labeling, we found that many tweets consisted of quotes, so we created a separate category for them, and asked the researchers to fix their labels accordingly.

[2]https://ifttt.com
[3]https://www.fitbit.com

Table 1: Examples of messages in each category

Spam	"Fellas need a mix 2 get ur lady in the mood heres a mix to help u succeed. [_URL]"
	"@X Listen To My Lul Song XXX Hot! Download And Share [_URL]"
	"Wow! another great item; available on eBay [_URL]"
	"WOW! No Cost Traffic For Your Website Home Based Business Blog Click Here Now Please #retweet [_URL]"
App-generated	"4 tweeps unfollowed (goodbye!) me in the past week. Thank you [_URL]."
	"New week; new tweets; new stats. 2 followers; 3 unfollowers. Via good old [_URL]."
	"August 28; 201X #Fitbit activity: 11XXX steps taken; 5.XX miles walked/ran; and 2XXX calories burned."
	"Your key planet Venus is now moving through your 12th House of... More for Libra [_URL]"
Quote	"Either you run the day or the day runs you. - Jim Rohn"
	"You're only as good as the people you hire. - Ray Kroc"
	"Do you want to know who you are? Don't ask. Act! Action will delineate and define you. - Thomas Jefferson"
	"Problems are only opportunities in work clothes. - Henry J. Kaiser"
Normal	"If you could ask a business consultant any question; what would you ask?"
	"Brendan Rodgers: Liverpool boss has no plans to leave club [_URL]"
	"RT @X: Thank you XX for last night. Hope you all enjoyed the show; you've always been lovely to us."
	"RT @X: Puppy caught eating paper decides killing the witness is the only way out [_URL]

Normal: A message that seems normal and has become popular (trending) because of its content. Examples are memes about current news and links to interesting reads, videos, photos, etc.
Unknown: If the coders were unsure about the category of a message, they could choose 'Unknown.' However, researchers were advised to try not to choose this option.

Table 1 provides some examples of manually labeled messages. Table 2 shows the size of each category after labeling all 5,000 clusters of similar messages. It also shows the number of tweets in each category. In total, by labeling these 5,000 groups, we obtained labels for 1,277,833 tweets. As you can see, clusters of similar messages are almost evenly labeled as normal (44%) and spam (42%). Most tweets though are labeled as normal (38%) because groups labeled as normal include more tweets. While only about 8% of groups are labeled as app, they include about 33% of tweets.

Labeling the tweets also can be utilized to classify users into two groups of spam and benign users. Table 2 shows the amount of unique users identified in each category. The total amount of unique users in all the groups is 66,788. However, some users appear in multiple categories. For users who appear in both spam and normal categories, three possible explanations are: First, spam accounts may try to emulate normal users to avoid suspension by Twitter; second, accounts may have been compromised, and, therefore, some posts on their timeline are benign while others are spam; and, third, their tweets are mislabeled.

Another interesting observation is that the amount of tweets in the normal category is only about 5% and 10% more than that in the spam and app categories, respectively, while the number of unique users in the normal category is almost 4 and 6 times more than that in the spam and app categories. These considerable differences indicate that spam accounts or campaigns are responsible for larger clusters that repeatedly post similar spam messages.

6.3 Identifying Parties of Interest

The probabilistic table representing the parties of interest is generated using the labeled dataset. As it was explained in Section 2, for each cluster of similar messages, POISED computes the probabilities of messages in that group being posted in each of the communities of interest. As an example, let us assume that in a neighborhood with three communities, the topic detection algorithm has found eight topics of interest and four-gram analysis has identified ten clusters of similar messages. The probabilistic table will then include ten rows and eight columns corresponding to groups and topics, respectively. The table entry for row i and column j is the probability that the messages in group i is being observed in communities with an interest in topic j.

Ideally, the probabilistic model should be built on the whole data of a social network. However, we cannot perform topic modeling on all data because of the resource constraints imposed by the use of MALLET. Instead, the topic detection is run and a probabilistic table is generated for each neighborhood separately. Each neighborhood includes multiple communities. We show that even while performing the analysis locally on each neighborhood, POISED detects spam messages effectively. This technique of running the analysis on neighborhoods can also help scaling our approach, so that it can be applied on much larger datasets. For example, Twitter can divide its large dataset into a few partitions that are then independently analyzed.

A probabilistic table for a neighborhood includes the clusters of similar messages that are posted in that neighborhood. Also, if messages of one large group are observed in multiple neighborhoods, then the probabilities for this group is computed separately and listed in the probabilistic table of every neighborhood. As a result, the size of probabilistic tables varies for each neighborhood. Some include thousands of clusters of similar messages while others include only a few of them. Because of the lack of enough observations in some neighborhoods, we did not run POISED on neighborhoods with less than 10 benign or 10 spam clusters of similar messages.

Therefore, our 'ground-truth' dataset for testing the second hypothesis includes the data for 202 neighborhoods with 2,896 clusters of similar messages and close to 1.3M tweets generated by more than 64K users.

Table 2 summarizes some statistics on this dataset. It shows that the number of spam groups and tweets in the ground-truth dataset are about 12% and 10% less than those in the 'labeled' dataset. In contrast, the number of app groups and tweets in the 'ground-truth' dataset are about 2% and 8% more than those in the 'labeled' dataset.

Table 2: Statistics for the manually labeled and ground-truth datasets

	Spam	App	Quote	Normal	Unknown
Labeled Dataset (300 neighborhoods)					
No. of groups	2,110 (42.2%)	376 (7.5%)	335 (6.7%)	2,178 (43.6%)	1 (0%)
No. of tweets	344,540 (27%)	416,099 (32.5%)	34,138 (2.7%)	482,975 (37.8%)	81 (0%)
No. of users	13,179	9,740	3,819	55,473	1
Ground-truth Dataset (202 neighborhoods)					
No. of groups	854 (30%)	274 (9%)	260 (9%)	1508 (52%)	
No. of tweets	168,181 (17%)	408,395 (40%)	32,607 (3%)	408,340 (40%)	
No. of users	12,504	9,614	3,815	55,219	

6.4 Classification on Parties of Interest

We employed machine learning to detect spam messages. The features are the list of topics in a neighborhood that are automatically detected by the topic modeling algorithm. An observation indicates the parties of interest represented by a row in the probabilistic table that includes the probabilities that a group of similar messages has been observed in communities interested in each of the topics. In addition to these features, we also consider the number of users per total number of messages in a group as a feature. This feature aims at capturing if a message is posted by several users or only by a small number of them.

The class of a group is the label in the ground-truth dataset. We define a class as a binary variable: 'benign' or 'spam'. While five categories were defined for manual labeling, for evaluation, we examined different combinations of these categories: *Comb. 1*: $Spam = \{spam\}$, and $Benign = \{normal, quote, app\}$, *Comb. 2*: $Spam = \{spam\}$, and $Benign = \{normal\}$ and *Comb. 3*: $Spam = \{spam, app\}$, and $Benign = \{normal, quote\}$. Later, we show that regardless of the combination, 'spam' messages are detected with high accuracy. The distinction between spam and benign messages can be specified by a policy and fed to POISED as a parameter.

The datasets are not balanced, *i.e.,* the number of spam and benign messages is not equal. We overcome this limitation by using a well-known over-sampling technique called SMOTE in which the minority class is over-sampled by creating "synthetic" examples [15].

To assess the effectiveness of our spam detection algorithm, we use the standard information retrieval metrics including recall, precision, F1-score, and accuracy.

After creating all the probabilistic models over the ground-truth dataset, we applied k-fold cross validation on each of the 202 neighborhoods separately. Then, we averaged their measures. We tested with three values of $k = \{3, 5, 10\}$ and found very similar results. Since $k = 10$ is one of the most common practices [10, 41, 66], we report the results for that value. We also tested several classification algorithms including Naive Bayes, SVM and Random Forest and all provide similar results.

Table 3 shows the results provided by SVM on different observation combinations. Since most groups of messages are labeled as 'spam' or 'normal', these combinations do not highly impact the results of the classifier. The results suggest that with high precision (91%) and recall (93%) our classifier successfully detects spam messages.

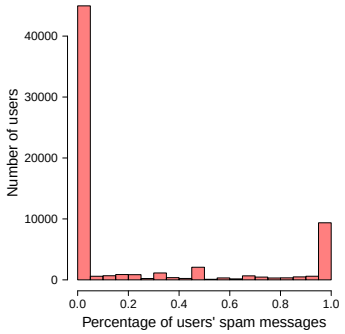


Figure 9: Percentage of spam posted by individual users.

6.5 Spam Accounts in Labeled Dataset

While some users only posted either benign or spam messages, some users have posted messages from both classes. Figure 9 indicates the histogram of spam message percentages for all users in the labeled dataset. It illustrates two main clusters of users who either post benign or spam content. This confirms that by examining the distribution of an account's messages, it is possible to label an account as benign or spam.

To identify spam accounts, we compute s, the percentage of spam messages to the total number of messages that a user has posted. A user is identified as a spam account if $s \geq \tau$. τ is a parameter that can be configured based on the dataset and the desired or accepted false positive values for the system. We set $\tau = 0.4$ and assume that if more than 40 percentage of the messages posted by an account are spam messages, then this account is more likely a spam account.

Finally, our labeled dataset includes 15,055 (23%) spam accounts and 49,675 (77%) honest users. The high percentage of spam accounts may be due to our selection of larger groups of similar messages as a labeled dataset, which may relatively include more spam messages compared to the whole Twitter data.

6.6 Comparison with state of the art systems

We compare POISED to three state-of-the-art systems that have been proposed by the research community in the past: SpamDetector [75], which is a system that detects fake accounts on social networks by examining characteristics of these profiles (*e.g.,* the fraction of messages posted that contain a URL), COMPA [27], which detects social network accounts that have been compromised by learning the typical behavior of an account and flagging any

Table 3: The performance of POISED as well as state of the art systems. The +/- values indicate standard error.

System	Label Combination	Accuracy	F1-score	Precision	Recall
POISED	Comb. 1	0.89 (+/- 0.02)	0.90 (+/- 0.02)	0.85 (+/- 0.02)	0.95 (+/- 0.02)
POISED	Comb. 2	0.90 (+/- 0.02)	0.91 (+/- 0.02)	0.87 (+/- 0.02)	0.95 (+/- 0.02)
POISED	Comb. 3	0.90 (+/- 0.02)	0.90 (+/- 0.02)	0.91 (+/- 0.02)	0.93 (+/- 0.02)
SpamDetector [75]	Comb. 3	0.67	0.20	0.21	0.20
COMPA [27]	Comb. 3	NA	0.55	1.0	0.38
BotOrNot [25]	Comb. 3, thr=0.8	0.76	0.07	0.36	0.04

activity that deviates from that behavior as a possible compromise, and BotOrNot [25, 29], which leverages more than one thousand features to evaluate if a Twitter account exhibits similarity to the known characteristics of social bots. We could not compare POISED with a couple of more recent work due to either the difficulty in obtaining their systems, or not being applicable on Twitter data. We discuss them in more details in Section 7.

Note that the threat model tackled by our approach is much broader than the one that these systems focused on: we aim to detect any malicious message regardless whether it was posted by a fake account or by a compromised one, while previous approaches only focused on one of these two categories. Because of this reason, our results in Table 3 show that our system outperforms both SpamDetector and COMPA, as well as BotOrNot.

SpamDetector and COMPA were developed as part of previous work by some of the authors of this paper, therefore we had access to their source code. In the case of SpamDetector, we performed a 10-fold cross validation on our labeled dataset using Random Forest as a classification algorithm. For COMPA, performing a 10-fold cross validation would not make sense, since this system does not take into account two classes of spam or benign accounts, but rather learns the typical behavior of an account and determines whether new messages that an account sends are malicious or not. Therefore, in this experiment we used COMPA to learn the typical behavior of the accounts that sent spam in our ground-truth dataset, and determine whether the spam messages that they sent were the consequence of a compromise.

Table 3 illustrates that our system outperforms both COMPA and SpamDetector considerably. The perfect precision reported by COMPA is an artifact of the fact that we only tested that system on malicious accounts (which is also the reason why we could not calculate accuracy), but the low recall of 0.38 (possibly due to the fact that only a minority of the accounts in our labeled dataset were compromised and not just fake) shows that POISED is a better candidate to fight the problem at hand.

We called the BotOrNot API for all the users in our labeled dataset. The classification system of BotOrNot is based on more than 1,000 features extracted from interaction patterns and content. When testing on a user, it returns the probability of that user being a social bot (Sybil account). To label users as bots, we picked multiple values, $\{0.7, 0.8, 0.9\}$, as a threshold. Here, we reported the values of measures for 0.8. We ran BotOrNot in February of 2016. Since both tools were run relatively close to each other, 60550 out of 63600 accounts still existed and were accessible for BotOrNot. In all cases, POISED outperforms BotOrNot. Overall, the precision of BotOrNot is around 40% while the recall is as low as 0.03.

6.7 Missed Spam Messages and Accounts

All of the above systems are relying on classifiers that find the abnormalities based on some features that can be adopted over time. The systems that we compared against POISED are either based on specific characteristics of the accounts under scrutiny (SpamDetector, BotOrNot) or look for changes in the behavior of an account that might be indicative of a compromise (COMPA). Given these peculiarities, these systems can only detect certain types of spam. In contrast, the feature set in POISED is adopted over time by detecting parties of interest and this makes its detection more comprehensive. In addition, POISED uses the main characteristic of spam messages, *i.e.,* their need to propagate in large-scale campaigns, which can not easily be mitigated by the attackers, who, by doing so, would directly affect the efficiency of their campaigns.

We further manually examined a sample of 'spam' accounts that are not detected by other approaches, to understand how POISED allows to improve the detection of spam over previous work.

Some of the users not detected by SpamDetector were users aggressively advertising products, while others were bots only tweeting about a specific hashtag. Also these accounts have multiple bots in their friends. We believe that these bots were able to evade the fairly simple threat model of SpamDetector, but were caught by the statistical models of POISED.

Some of the missed spam accounts by COMPA were users who retweeted quotes to appear legitimate, but also posted spam from their blog. Again we found multiple bots in their friends. We believe that these accounts were not compromised, but rather bot accounts, and were therefore outside the threat model used by COMPA. Both COMPA and SpamDetector did not identify users posting automated content from applications.

BotOrNot, on the other hand, falsely detected some benign users as bots, possibly because of their high follower-to-friend ratio and low count of tweets. It also missed some accounts that were detected by POISED, that have since been suspended by Twitter, possibly for spamming. These examples show that POISED is able to identify a broader category of spammers than previous systems, and is therefore more effective in fighting this problem.

6.8 Early Spam Detection

POISED is more effective if it can identify spam messages early on, before they are fully distributed throughout the network. We investigated the impact of "early detection" on the performance of POISED. We implemented a simulation, where at the probabilistic table creation phase, for each spam observation, the propagation probabilities are only obtained for some percentage of communities,

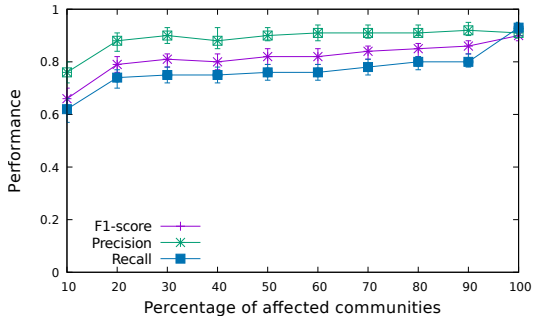


Figure 10: Effective in early detection of spam messages

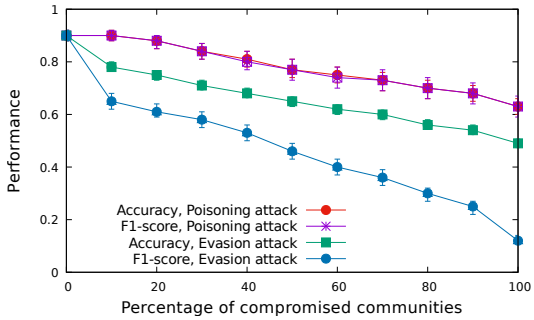


Figure 11: POISED **vs. adversarial machine learning attacks**

and assumes that the message has not been observed in the remaining communities. We ran experiments for different percentages of communities, 10%-100%, where communities are picked randomly. For each percentage, we repeated the experiments three times.

Figure 10 indicates that POISED is effective in detecting spam messages at the early stage when they have only propagated through 30% of the communities with 90% precision and 75% recall.

6.9 Adversarial Machine Learning Attacks

To evade the protection, adversaries actively manipulate data to make the classifier produce false negatives [4, 21, 34, 46, 52, 58, 71]. These attacks will usually target two different stages of classification: 1) at the training phase, where an adversary may attempt to mislead the classifier by "poisoning" its training data with carefully designed attacks [70], or 2) at the testing phase where an adversary may attempt to evade a deployed system by carefully manipulating attack samples [5]. These attacks are called *poisoning* and *evasion* attacks respectively. We investigated the robustness of POISED against both of these adversarial settings to understand to what extent our classifier can resist the targeted attacks.

In both attacks, an adversary attempts to make the propagation of her spam message be as similar as possible to that of a benign message in the whole network.

For that, we assume that the adversary has the ability to create sybil accounts, establish connections with honest users and pretend to share the same interest as target communities by sending topical messages. As the result of these malicious activities, we assume that the adversary obtains the knowledge of: 1) the message counts in each of the communities of interest, and 2) the number of users who have posted those messages in each of those communities of interest. Moreover, we assume that the number of fake or compromised accounts is equivalent to the number of users (re-)posting that specific benign message because this is one of the features used in the classifier.

However, we assume that in non-compromised communities, the adversary is unable to control any of these variables. For example, let us take a network with 200 communities, where the attacker has compromised 50 communities. She observes that some similar benign messages are always posted in 30 of these communities. Using this information, she posts her malicious content with a similar distribution in those same 30 communities. However, since she has no control over 150 of the 200 communities, her messages most likely propagate differently than benign messages through the network.

We simulated the attacks by randomly picking some percentage of communities as *compromised* by the adversary.

The poisoning attack is performed during the training phase. At this phase, the adversary deliberately interferes with both the topic detection and community detection algorithms. She joins communities in order to modify the network and the community structures. In these now compromised communities, she posts messages resembling the propagation of benign messages in the compromised communities, therefore modifying the probability distribution of topics in these communities. The propagation probabilities for this spam message is actually the combination of probabilities for the chosen benign message in the *compromised* communities and those of the spam message in the *non-compromised* communities. However, since the observations in the training set are (manually) labeled, the spam messages should still be correctly labeled as spam. The classifier might not be able to accurately distinguish and detect either spam or benign messages because the parties of interest for benign and spam messages are partially identical.

The evasion attack is performed during the testing phase. The training set is not polluted while the testing set includes the instances of spam messages that attempt to resemble the probabilistic propagation of benign messages. In this attack, we assume that the attacker has partial knowledge about the model built on the training set, and modifies the probability distribution of her messages in compromised communities, as explained previously at the poisoning attack.

For both attacks, in each of the neighborhoods and for each percentage of compromised communities, we ran the simulations three times and averaged all the results. In the case of the poisoning attack, the observations of all the spam messages in the ground-truth are modified based on the attack to represent simulated spam messages, while the observations for benign messages remain unmodified. To evaluate the attack, we performed 10-fold cross-validation on a balanced dataset generated by SMOTE algorithm [15].

In the case of the evasion attack, the testing set includes all simulated spam messages that are generated based on the attack. It also includes a random set of x benign messages where x is equal to the number of spam messages in the testing set. The training set is the entire ground-truth dataset excluding those benign messages appearing in the testing set.

Figure 11 shows the impact of these attacks on the performance of POISED when the percentage of compromised communities increases from 0 to 100. As it can be seen in the results, by increasing the percentage of compromised communities, the performance gradually decreases in both attacks. In the poisoning attack, F-1 decreases from 90% to 65%, while in the evasion attack, it decreases from 90% to 15% when the number of compromised communities increases from 0 to 100%. POISED is more robust to the poisoning attack because simulated spam messages and their propagation probabilities in non-compromised communities are observed in the training set and the classifier has learned from them.

These results suggest that the attacker needs to have a great knowledge about the network to highly impact the performance of the classifier. For example, even if 30% of the network is compromised, the precision and recall remain at 82% and 87% in the case of a poisoning attack, and at 75% and 52% in the case of an evasion attack. Moreover, we argue that the poisoning attack is a more realistic attack because, in practice, the ground-truth dataset gets updated over time and the training set most probably includes spam messages generated by the adversary.

7 RELATED WORK

POISED is the first system able to detect spam on Twitter by looking at the differences in which legitimate and malicious messages propagate through the network. In the following, we discuss previous work in the area of spam detection on social networks. Broadly speaking we identify two types of approaches: those that look at identifying malicious messages and those that look at flagging malicious accounts. We then revise the academic literature that studied message propagation on online social networks.

Malicious Messages Detection. Identifying spam messages on social networks has been extensively studied [47, 78, 80, 90, 92, 94, 95]. Yardi *et al.* [94] studied Twitter spammers who abuse trending topics. Analyzing URLs in messages is another method employed to detect malicious messages [47, 80]. Monarch [80] is a system for crawling URLs spread in social network and identifying malicious messages and compromised accounts. Lee and Kim [47] also proposed WarningBird, a system that analyzes correlated redirection chains of URLs in a number of URLs posted on Twitter to identify malicious tweets. Some researchers have proposed offline spam analysis to identify large-scale social spam campaigns [32, 36]. They mostly apply clustering algorithms based on URL blacklisting on a complete set of messages. Xu *et al.* [92] presented an early warning worm detection system that monitors the behavior of users to collect suspicious worm propagation evidences.

In summary, these works concentrate on limited aspects of the spam detection problem, such as messages URL analysis, user behavior analysis, offline spam analysis, etc. In POISED, we provide a generic solution that detects spam messages from their propagation patterns through communities of interest, regardless of their content.

Malicious Accounts Detection. Today, normal users in popular social networks are increasingly becoming the target of attackers. Many research works investigated this problem and proposed various solutions for this challenge [3, 8, 9, 11, 12, 22, 28, 29, 32, 75, 91]. COMPA [28] is a system that detects compromised Twitter accounts based on their behaviors over time. The authors showed that normal users have almost stable habits over time, unlike compromised users who likely show anomalous habits. Liu *et al.* [51] calculated user topics with LDA, and then employed supervised learning to identify spammers based on topics of discussion. Cai *et al.* [11] presented a machine learning-based platform to detect Sybil attacks in social networks. They split a social network into communities, and tried to identify communities that connect in an unnatural or inconsistent way with the rest of the social network. SybilInfer [22] detects compromised accounts using a Bayesian Inference approach. Stringhini *et al.* [75] investigated spammers' behavior by creating a set of honey-profiles on popular social networks. By studying spammers' characteristics, they introduced a spam detection tool. Link Farming in Twitter where spammers acquire large number of follower links has been investigated by Ghosh *et al.* [33]. By analyzing over 40,000 spammer accounts, they discovered that a majority of farmed links comes from a small number of legitimate and highly active users. Wang *et al.* [87] analyzed user click patterns to create user profiles and identify fake accounts using both supervised and unsupervised learning. Viswanath *et al.* [85] applied Principal Components Analysis (PCA) to find patterns among features extracted from spam accounts. Cao *et al.* proposed SynchroTrap [13], a detection system that clusters malicious accounts according to their actions and the time at which they are made. EvilCohort [76] is a system that identifies sets of social network accounts used by botnets, by looking at communities of accounts that are accessed by a common set of IP addresses.

These works aim to detect a particular type of malicious user (*e.g.,* compromised accounts or fake accounts). Instead, we propose a comprehensive spam detection system independent from the type of malicious user spreading it.

Message Propagation. POISED is the first system that proposes to detect spam on Twitter by looking at how legitimate and malicious messages spread on the social network. Previous work, however, looked at message propagation on social networks for other purposes.

Ye and Wu [95] studied propagation patterns of general messages and breaking news in Twitter. They inspected a massive number of messages collected from 700K users. Moreover, they evaluated different social influences by analyzing their changes over time, and how they correlate with each other. By analyzing Twitter hashtags, Weng *et al.* [90] showed that network communities can help predicting viral memes. In summary, the popularity of a meme can be predicted by quantifying its early spreading pattern in terms of community concentration: the more communities a meme permeates, the more viral it is. Nematzadeh *et al.* [59] demonstrated that strong communities with high modularity can facilitate global diffusion by enhancing local, intra-community spreading. Through a simulation, Mezzour *et al.* [55] showed how the diffusion of messages by hacked accounts differs from normal accounts. Similar to these works, we analyze how messages propagate in social networks. We combine this to learn parties of interest and detect spam messages.

8 DISCUSSION

Data Collection. Not having access to the whole Twitter data, including the users' data and Twitter network, imposes some constraints to our approach. For example, it may affect the quality of the communities of interest as well as the groups of similar messages. Nonetheless, with these limitations, POISED performed well in detecting diverse spam messages.

Complexity and Scalability. POISED is practical even at the scale of Twitter. Here, we discuss the complexity of the multiple phases of our approach. The time complexity for Infomap is estimated at $O(m)$, where m is the number of nodes. It can classify millions of nodes in minutes [2, 44].

The topic detection algorithm, LDA, has a complexity of $O(NKV)$, where N, K, and V are the number of documents, topics and words in the vocabulary, respectively. The efficiency of LDA can be improved with the use of heuristics, *e.g.,* by running it on each neighborhood independently, decreasing the number of documents by having more tweets per document, and specifying a lower number of topics. Recent work has also explored multiple approaches for increasing the performance of LDA [35, 57, 65, 73, 88], which can be applied to POISED.

Identifying groups of similar messages is a string searching problem. POISED classifies messages based on four-gram matches. The length of tweets is short, and each of them only contains a few consecutive four-grams. The time complexity and memory complexity for four-gram analysis are $O(N^2M^2)$, where N is the number of messages and M is the maximum size of a message (*i.e.,* 140 characters in the case of Twitter). However, the analysis can be optimized from $O(N^2M^2)$ to close to linear in the number of similar groups returned [17, 24, 40].

In our experiments, running the four-gram analysis using a commodity desktop took approximately an hour. Running LDA on 300 neighborhoods, with 15M tweets, took nine hours on a commodity desktop. Each neighborhood analysis can be run independently, and thus, can be parallelized in order to scale. For 500M tweets a day, which consists in the daily average on Twitter, we estimate that it would require approximately 150 machines to run our distributed analysis in two hours.

Finally, POISED successfully detects spam messages in neighborhoods. Therefore, to increase the efficiency, POISED can be run on some partitions of networks. Furthermore, the most demanding steps are topic detection and message matching, which prepare the training data for the probabilistic model. These can be run offline and less frequently. The SVM classifier itself is highly efficient.

Live implementation. Although POISED involves many operations, testing new messages on a live system would require fewer steps. The actual process involved in identifying messages as spam would be to 1) establish groups of similar messages, 2) observe their propagation through parties of interest, and 3) test them on the probabilistic model that is obtained in advance. Twitter can collect user reports over time and use them as the ground-truth to build the probabilistic model. The communities of interest must be computed regularly, as the network likely evolves through time.

Ground-truth dataset. For online social networks such as Twitter and Facebook, obtaining a ground-truth dataset requires minimal effort; they already have some mechanisms in place for their users to report malicious behavior. Over time, following the network evolution, they can update or add to their ground-truth dataset. Note that our results show that POISED is able to detect unseen malicious content.

Limitations. Similar to other spam detection systems [27, 74], POISED groups similar messages before applying the probabilistic model to classify spam. An attacker aware of this feature could evade the four-gram analysis used to identify messages similarity. However, in that case, the similarity analysis could be extended with more complex similarity measures [27].

Our approach requires a minimum number of users and messages to form communities of interest. A malicious user with a small number of connections might be able to evade our detection. However, this goes directly against the interest of malicious users, who want to reach as many victims as possible.

9 CONCLUSIONS AND FUTURE WORK

In this paper, we presented POISED, a novel and general approach for detecting social network spam based on its diffusion through parties of interest. First, we established that members of networked communities share common topics of interest distinct from other surrounding communities. We then built a probabilistic model that detects spam based on the dissemination of messages through communities of interest with high efficiency. We also showed that POISED outperforms other spam detection systems proposed in recent work, while its threat model is also more general. Moreover, we showed how POISED is effective in the early detection of spam messages and how it is resilient against two well-known adversarial machine learning attacks.

In this paper, we assumed that users are only member of one particular community. A possible future work is to explore detecting overlapping communities of interest. We can also explore the combination of our framework to existing systems that analyze the characteristics of user accounts or messages to detect spam. Finally, another future work is to employ and test POISED on other online social networks.

ACKNOWLEDGEMENTS

This research was supported by the EPSRC under Grant N008448 and by the European Commission as part of the ENCASE project (H2020-MSCA RISE of the European Union under GA number 691025). It is also in part supported by ESET and FRQNT.

This material is also based upon work supported by the National Science Foundation under Awards No. DGE-1623246, and CNS-1408632, as well as by Office of Naval Research under Award No. N00014-17-1-2011, and in part is supported by Google. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the above agencies. Also, this material is based on research sponsored by DARPA under agreement number FA8750-15-2-0084. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

REFERENCES

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. 2010. Link communities reveal multiscale complexity in networks. *Nature* 466, 7307 (2010), 761–764.

[2] Rodrigo Aldecoa and Ignacio Marín. 2013. Exploring the limits of community detection strategies in complex networks. *Scientific reports* 3 (2013).

[3] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. 2010. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, Vol. 6. 12.

[4] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. 2011. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International Workshop on Multiple Classifier Systems*. Springer, 350–359.

[5] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 387–402.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3 (2003), 993–1022.

[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008.

[8] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Lería, Jose Lorenzo, Matei Ripeanu, and Konstantin Beznosov. 2015. Integro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs.. In *NDSS*, Vol. 15. 8–11.

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

[10] Ulisses M Braga-Neto and Edward R Dougherty. 2004. Is cross-validation valid for small-sample microarray classification? *Bioinformatics* 20, 3 (2004), 374–380.

[11] Zhuhua Cai and Christopher Jermaine. 2012. The latent community model for detecting sybil attacks in social networks. In *Proc. NDSS*.

[12] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 15–15.

[13] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 477–488.

[14] Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean Birds: Detecting Aggression and Bullying on Twitter. In *International ACM Web Science Conference (WebSci)*.

[15] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[16] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and Mobility: User Movement in Location-based Social Networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA.

[17] Ondrej Chum, James Philbin, Andrew Zisserman, et al. 2008. Near Duplicate Image Detection: min-Hash and tf-idf Weighting.. In *BMVC*, Vol. 810. 812–815.

[18] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E* 70, 6 (2004), 066111.

[19] B Csákány. 1981. Homogeneity and completeness. In *Fundamentals of Computation Theory*. Springer, 81–89.

[20] Mary J Culnan, Patrick J McHugh, and Jesus I Zubillaga. 2010. How large US companies can use Twitter and other social media to gain business value. *MIS Quarterly Executive* 9, 4 (2010), 243–259.

[21] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. 2004. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 99–108.

[22] George Danezis and Prateek Mittal. 2009. SybilInfer: Detecting Sybil Nodes using Social Networks.. In *NDSS*. San Diego, CA.

[23] Stephen Dann. 2010. Twitter content classification. *First Monday* 15, 12 (2010).

[24] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 271–280.

[25] Clayton A Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. BotOrNot: A System to Evaluate Social Bots. *arXiv preprint arXiv:1602.00975* (2016).

[26] Eric Eaton and Rachael Mansbach. 2012. A Spin-Glass Model for Semi-Supervised Community Detection.. In *AAAI*. Citeseer.

[27] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2013. COMPA: Detecting Compromised Accounts on Social Networks.. In *NDSS*.

[28] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2015. Towards Detecting Compromised Accounts on Social Networks. *Transactions on Dependable and Secure Computing (TDSC)* (2015).

[29] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2014. The rise of social bots. *arXiv preprint arXiv:1407.5225* (2014).

[30] Chris Fleizach, Geoffrey M Voelker, and Stefan Savage. 2007. Slicing spam with occam's razor. In *CEAS*.

[31] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. 2012. Towards Online Spam Filtering in Social Networks.. In *NDSS*.

[32] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. 2010. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 35–47.

[33] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. 2012. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 61–70.

[34] Amir Globerson and Sam Roweis. 2006. Nightmare at Test Time: Robust Learning by Feature Deletion. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*. ACM, New York, NY, USA.

[35] Ryan Gomes, Max Welling, and Pietro Perona. 2008. Memory bounded inference in topic models. In *Proceedings of the 25th international conference on Machine learning*. ACM, 344–351.

[36] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. 2010. @ spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 27–37.

[37] Liangjie Hong and Brian D Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*. ACM, 80–88.

[38] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. 2007. Social phishing. *Commun. ACM* 50, 10 (2007), 94–100.

[39] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*. ACM, 56–65.

[40] Chulyun Kim and Kyuseok Shim. 2011. Text: Automatic template extraction from heterogeneous web pages. *IEEE Transactions on knowledge and data Engineering* 23, 4 (2011), 612–626.

[41] Ron Kohavi. 1995. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2 (IJCAI'95)*. Morgan Kaufmann Publishers Inc.

[42] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a Social Network or a News Media?. In *Proceedings of the 19th International Conference on World Wide Web*. ACM.

[43] Andrea Lancichinetti and Santo Fortunato. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E* 80, 1 (2009), 016118.

[44] Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: a comparative analysis. *Physical review E* 80, 5 (2009), 056117.

[45] A Lancichinetti, MI Sirer, JX Wang, D Acuna, K KÃűrding, and LAN Amaral. 2015. High-reproducibility and high-accuracy method for automated topic classification. *Physical Review X* 5 (29 JAN 2015), 011007. https://doi.org/10.1103/PhysRevX.5.011007

[46] Pavel Laskov and Richard Lippmann. 2010. Machine learning in adversarial environments. *Machine learning* 81, 2 (2010), 115–119.

[47] Sangho Lee and Jong Kim. 2012. WarningBird: Detecting Suspicious URLs in Twitter Stream.. In *NDSS*.

[48] Kristina Lerman and Rumi Ghosh. 2010. Information contagion: An empirical study of the spread of news on Digg and Twitter social networks. *ICWSM* 10 (2010), 90–97.

[49] Kwan Hui Lim and Amitava Datta. 2012. Finding twitter communities with common interests using following links of celebrities. In *Proceedings of the 3rd international workshop on Modeling social media*. ACM, 25–32.

[50] Changchang Liu, Peng Gao, Matthew Wright, and Prateek Mittal. 2015. Exploiting Temporal Dynamics in Sybil Defenses. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[51] Linqing Liu, Yao Lu, Ye Luo, Renxian Zhang, Laurent Itti, and Jianwei Lu. 2016. Detecting" Smart" Spammers On Social Network: A Topic Model Approach. *arXiv preprint arXiv:1604.08504* (2016).

[52] Daniel Lowd and Christopher Meek. 2005. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 641–647.

[53] Andrew K McCallum. 2002. {MALLET: A Machine Learning for Language Toolkit}. http://mallet.cs.umass.edu/. (2002).

[54] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001), 415–444.

[55] Ghita Mezzour and Kathleen M Carley. 2014. Spam diffusion in a social network initiated by hacked e–mail accounts. *International Journal of Security and Networks* 9, 3 (2014), 144–153.

[56] Peter J Mucha, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. 2010. Community structure in time-dependent, multiscale, and multiplex networks. *science* 328, 5980 (2010), 876–878.

[57] Ramesh Nallapati, William Cohen, and John Lafferty. 2007. Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. IEEE, 349–354.

[58] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. 2008. Exploiting Machine Learning to Subvert Your Spam Filter. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*. USENIX Association, Berkeley, CA, USA.

[59] Azadeh Nematzadeh, Emilio Ferrara, Alessandro Flammini, and Yong-Yeol Ahn. 2014. Optimal network modularity for information diffusion. *Physical review letters* 113, 8 (2014), 088701.

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

[61] Mark EJ Newman and Juyong Park. 2003. Why social networks are different from other types of networks. *Physical Review E* 68, 3 (2003), 036122.

[62] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (June 2005), 814–818.

[63] Patrick O Perry and Patrick J Wolfe. 2012. Null models for network data. *arXiv preprint arXiv:1201.5871* (2012).

[64] Pascal Pons and Matthieu Latapy. 2005. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*. Springer, 284–293.

[65] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 569–577.

[66] Payam Refaeilzadeh, Lei Tang, and Huan Liu. 2009. Cross-validation. In *Encyclopedia of database systems*. Springer, 532–538.

[67] Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM* (2011).

[68] Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure.. In *EMNLP-CoNLL*, Vol. 7. 410–420.

[69] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105, 4 (2008), 1118–1123.

[70] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 1–14.

[71] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-world Exploits. In *Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15)*. USENIX Association, Berkeley, CA, USA, 1041–1056. http://dl.acm.org/citation.cfm?id=2831143.2831209

[72] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[73] Padhraic Smyth, Max Welling, and Arthur U Asuncion. 2009. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems*. 81–88.

[74] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna. 2012. Poultry Markets: On the Underground Economy of Twitter Followers. In *Proceedings of the Workshop on Online Social Network (WOSN)*. ACM, Helsinki, Finland.

[75] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2010. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 1–9.

[76] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2015. Evilcohort: detecting communities of malicious accounts on online services. In *USENIX Security Symposium*.

[77] Jessica Su and Sen Wu. 2013. Null Models For Social Networks. http://snap.stanford.edu/class/cs224w-2013/projects2013/cs224w-003-final.pdf. (2013).

[78] Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic-and author-controlled natural experiments on Twitter. *arXiv preprint arXiv:1405.1438* (2014).

[79] Lei Tang and Huan Liu. 2010. Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery* 2, 1 (2010).

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

[81] Kurt Thomas, Chris Grier, Dawn Song, and Vern Paxson. 2011. Suspended accounts in retrospect: an analysis of twitter spam. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 243–258.

[82] Oren Tsur, Adi Littman, and Ari Rappoport. 2013. Efficient Clustering of Short Messages into General Domains.. In *ICWSM*. Citeseer.

[83] Twitter. 2009. Reporting spam on Twitter. https://support.twitter.com/articles/64986. (2009).

[84] Twitter. 2015. Twitter usage. https://about.twitter.com/company. (2015).

[85] Bimal Viswanath, Muhammad Ahmad Bashir, Mark Crovella, Saikat Guha, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. 2014. Towards Detecting Anomalous User Behavior in Online Social Networks.. In *Usenix Security*, Vol. 14.

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

[87] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao. 2013. You Are How You Click: Clickstream Analysis for Sybil Detection.. In *Usenix Security*, Vol. 14.

[88] Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen, and Edward Y Chang. 2009. Plda: Parallel latent dirichlet allocation for large-scale applications. In *International Conference on Algorithmic Applications in Management*. Springer, 301–314.

[89] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 261–270.

[90] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. 2013. Virality prediction and community structure in social networks. *Scientific reports* 3 (2013).

[91] Baoning Wu, Vinay Goel, and Brian D Davison. 2006. Topical trustrank: Using topicality to combat web spam. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 63–72.

[92] Wei Xu, Fangfang Zhang, and Sencun Zhu. 2010. Toward worm detection in online social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 11–20.

[93] Chao Yang, Robert Chandler Harkreader, and Guofei Gu. 2011. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *Recent Advances in Intrusion Detection (RAID)*.

[94] Sarita Yardi, Daniel Romero, Grant Schoenebeck, et al. 2009. Detecting spam in a twitter network. *First Monday* 15, 1 (2009).

[95] Shaozhi Ye and S Felix Wu. 2010. *Measuring message propagation and social influence on Twitter*. Springer.