Towards Online Spam Filtering in Social Networks

Hongyu Gao
Northwestern University
Evanston, IL, USA
hygao@u.northwestern.edu

Yan Chen
Northwestern University
Evanston, IL, USA
ychen@northwestern.edu

Kathy Lee†
Northwestern University
Evanston, IL, USA
kml649@eecs.northwestern.edu

Diana Palsetia†
Northwestern University
Evanston, IL, USA
palsetia@u.northwestern.edu

Alok Choudhary†
Northwestern University
Evanston, IL, USA
choudhar@eecs.northwestern.edu

Abstract

Online social networks (OSNs) are extremely popular among Internet users. Unfortunately, in the wrong hands, they are also effective tools for executing spam campaigns. In this paper, we present an online spam filtering system that can be deployed as a component of the OSN platform to inspect messages generated by users in real-time. We propose to reconstruct spam messages into campaigns for classification rather than examine them individually. Although campaign identification has been used for offline spam analysis, we apply this technique to aid the online spam detection problem with sufficiently low overhead. Accordingly, our system adopts a set of novel features that effectively distinguish spam campaigns. It drops messages classified as "spam" before they reach the intended recipients, thus protecting them from various kinds of fraud. We evaluate the system using 187 million wall posts collected from Facebook and 17 million tweets collected from Twitter. In different parameter settings, the true positive rate reaches 80.9% while the false positive rate reaches 0.19% in the best case. In addition, it stays accurate for more than 9 months after the initial training phase. Once deployed, it can constantly secure the OSNs without the need for frequent re-training. Finally, tested on a server machine with eight cores (Xeon E5520 2.2Ghz) and 16GB memory, the system achieves an average throughput of 1580 messages/sec and an average processing latency of 21.5ms on the Facebook dataset.

1 Introduction

Online social networks (OSNs) are extremely popular collaboration and communication tools that have attracted millions of Internet users. Unfortunately, recent evidence shows that they can also be effective mechanisms for spreading attacks. Popular OSNs are increasingly becoming the target of phishing attacks launched from large botnets [1, 3]. Two recent studies [9, 10] have confirmed the existence of large-scale spam campaigns in Twitter and Facebook, respectively. Furthermore, the clickthrough rate of OSN spam is orders of magnitude higher than its email counterpart [10], indicating that users are more prone to trust spam messages from their friends in OSNs.

The OSN spam problem has already received attention from researchers. Meanwhile, email spam, a seemingly very similar problem, has been extensively studied for years. Unfortunately, the bulk of the existing solutions are not directly applicable, because of a series of distinct characteristics pertaining to the OSN spam. i) In any OSN, all messages, including spam, originate from accounts registered at the same site. In contrast, email spam is *not* necessarily sent from accounts registered at legitimate service providers. The widely used email server reputation based detection approaches [11, 19, 22] rely on the assumption that the spamming SMTP servers run on bot machines, and are thus inapplicable in OSNs. Realizing that this assumption is not always true , researchers have proposed to identify accounts signed up by spammers from legitimate email service providers [34]. ii) Spamming account identification is also the focus of the existing OSN spam detection work [14, 26, 31, 32]. However, the second characteristic of OSN spam is that the majority of spam messages come from compromised accounts [9, 10], rather than accounts created and exclusively controlled by spammers. It essentially means that spammers and legitimate users are sharing accounts. Thus, identifying spamming accounts is not sufficient to fight OSN spam. iii) Messages in OSNs, spam or not, are generally short. The perception that legitimate emails have variable size while spam tends to be small

no longer holds in OSNs. The third characteristic, along with the previous two, obsoletes many features that used to work well in supervised machine learning based detection approaches, which we briefly discuss in Section 3.

It is worth noting that offline OSN analysis works successfully reveal OSN spam messages [9, 10], but they are not designed as online detection tools, since they either have long lag-time or limited efficiency. We offer more detailed comparison in Section 7. Meanwhile, message content based detection approaches, such as the recently proposed campaign template inference [18], are expected to perform equally well if applied to OSNs. However, they implicitly require that *all* campaigns must be present in the labeled training set in order for good detection coverage. This requirement itself is hard to satisfy, not to mention the difficulty of fighting campaigns that emerge in the future. In addition, approaches based on malicious URL detection are also applicable [27]. Nonetheless, URL obfuscation techniques (*e.g.* , using "nevasubevu\t. blogs pot\t.\tco\tm (take out spaces)" instead of "nevasubevu.blogspot.com") make it difficult for any automated tool to recognize the embedded URLs in the first place.

In this paper, we present an online spam filtering system specifically designed for OSNs and can be deployed as a component of the OSN platform. After the initial training phase, it efficiently inspects the stream of user generated messages, immediately dropping those classified as spam before they reach the intended recipients. The system owns four desirable properties as an online filtering tool, which are: i) high accuracy, ii) no need for all campaigns to be present in the training set, iii) no need for frequent re-training , and iv) low latency.

The key insight is that we always seek to uncover the connection among all the messages by performing clustering on them, instead of directly inspecting each individual message without correlating it with others. The correlated spam messages form spam campaigns. Although the clustering approach has been used for offline spam analysis [4, 35], it is never used for online spam filtering due to its computational overhead. We leverage incremental clustering and parallelization to address this challenge. When a new message is generated, the system organizes it, along with all the previously observed messages, into clusters. The new message is then classified according to whether or not the cluster it resides in is a spam cluster, which is determined by all the messages in the same cluster collectively.

Accordingly, the classifier is trained on a set of features associated with message clusters instead of individual messages. We identified six such distinguishing features. They all stem from the spammers' need to push spam messages to as many recipients as possible. Meanwhile, these features grasp the *commonality* among spam campaigns, *e.g.* , they generate spam messages fast while lasting, rather than the speciality of any individual campaign, *e.g.* , word distribution or underlying templates. Hence, spam campaigns can still be detected even if they do not have any instance included in the training set. In addition, the features pertaining to campaigns cannot be easily maneuvered. For example, if the spammer slows down the speed of spam message generation, he has to pay the price of the reduced ability to affect potential victims. As a result, these features are persistent over time. After the initial training, the system can work online for a long period of time without the need for re-training. For the same reason, it is also harder for the spammers to craft their sending pattern to evade detection. We evaluate the system using 187 million wall posts collected from Facebook and 17 million tweets collected from Twitter. Experimental results show that the system yields 80.9% true positive rate and 0.19% false positive rate in the best parameter settings. It stays accurate for more than 9 months after the initial training phase, and achieves an average throughput of 1580 messages/sec and an average processing latency of 21.5ms on the Facebook dataset.

In summary, we frame our contributions as:

- We propose to use spam campaigns, instead of individual spam messages, as the objects for spam classification.
- We solve the challenge of reconstructing campaigns in real-time by adopting incremental clustering and parallelization.
- We identify six features that distinguish spam campaigns from legitimate message clusters in OSNs.
- We develop and evaluate an accurate and efficient system that can be easily deployed at the OSN server side to provide online spam filtering.

The rest of the paper is organized as follows. We first provide necessary background information and clarify what will be achieved in Section 2. After that, we present the six features to distinguish spam clusters in Section 3, illustrate the detailed system design in Section 4, and thoroughly evaluate its performance in Section 5. Next, we discuss possible attempts to evade the system in Section 6 and the related work in Section 7. Finally, we conclude the paper in Section 8.

2 Background and Goal

In this section, we provide necessary background information and how our system can be incorporated into the current OSN architecture. We also describe the dataset used in this study.

2.1 Background

All current OSNs adopt the client-server architecture. The OSN service provider acts as the controlling entity. It stores and manages all the content in the system. On the other hand, the content is generated by users spontaneously from the client side. The OSN service provider offers a rich set of well-defined interfaces through which the users can interact with others. Currently two popular ways of interaction exist. Facebook is representative of OSNs that adopt the interaction between a pair of sender and recipient as their primary way of interaction, although they also support other ways. Twitter is representative of OSNs that adopt broadcasting as their primary way of interaction.

Figure 1(a) illustrates a simplified OSN architecture. It only depicts the components that are related to message exchanging, while all other functionalities, *e.g.* , user authentication, video sharing and 3rd party applications, are omitted. In this simplified example, multiple users are interacting via the message posting and viewing interface. In Facebook-like OSNs, it represents the case that user A and B are posting messages to user C and D, respectively. In Twitter-like OSNs, it represents the case that user A and B are broadcasting messages to all the followers including user C and D, respectively, while other possible recipients are omitted for simplicity. In both cases, the service provider mediates all the interactions. The generated messages are first stored at the service provider's side, and will be relayed when the corresponding recipient signs in.

Unfortunately, all the content in OSNs is generated by users and is not necessarily legitimate. The posted messages could be spam. Note that although spam traditionally refers to massive, unsolicited campaigns trying to promote products or services, we do not restrict ourselves to this behavior alone. Rather, we use the term "spam" to refer to unsolicited campaigns that attempt to execute a variety of attacks, including but not restricted to: i) product advertisements, ii) phishing and iii) malware spreading. In the example of Figure 1(a), user A's account is compromised and sends a spam message to user C, trying to direct user C to some malicious website. Once user C signs in, the spam message will be displayed to him, exposing him to potential threats.

2.2 Goal

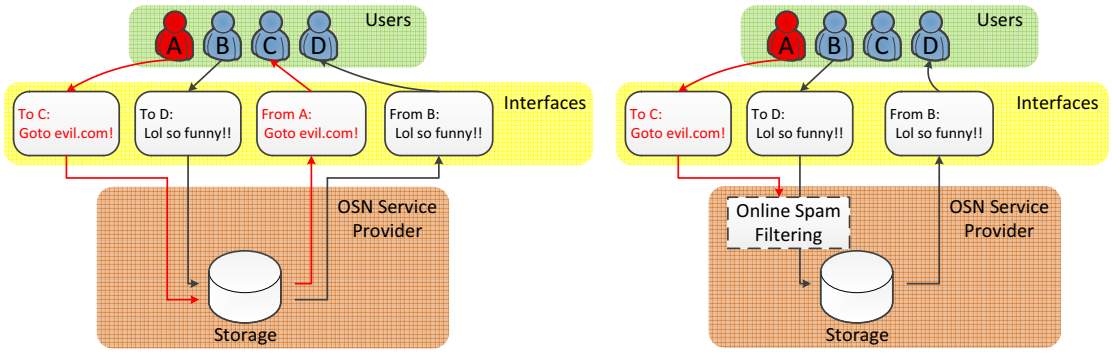
Our goal is to design an online spam filtering system that is deployed at the OSN service provider side, as Figure 1(b) shows. Once deployed, it inspects every message before rendering the message to the intended recipients and makes immediate decision on whether or not the message under inspection should be dropped. In this particular example, the message generated by user A is classified as spam and is thus dropped instantly. The message from user B is legitimate and is stored by the service provider. Later when user C and D sign in to the system, C will not see the dropped spam message.

We assume the spam message originates from a compromised account in the above example. We stress that this is *not* an assumption that our system relies on. Rather, our system can detect spam messages originating from both compromised accounts and spamming bots.

2.3 Dataset

Facebook and Twitter are two representative OSNs. We use data collected from both sites in the study. The Facebook dataset contains 187 million wall posts generated by roughly 3.5 million users in total, between January of 2008 and June of 2009 [29]. For the Twitter data collection, we first download trending topics, *i.e.* , popular topics, from the website *What the Trend* [2], which provides a regularly updated list of trending topics. We then download from Twitter all public tweets that contain the trending topics while the topics are still popular via Twitter APIs. For example, while the topic "MLB" is trending, we keep downloading all tweets that contain the word "MLB" from Twitter. For each tweet we also obtain the userID that generates it along with its friend number, *i.e.* , the number of users it follows. the Twitter dataset contains over 17 million tweets related to trending topics that were generated between June 1, 2011 and July 21, 2011. The primary form of communication in Facebook and Twitter is called "wall post" and "tweet", respectively. From now on, we use the term "message" to refer to both of them for the ease of description.

We need labeled spam and legitimate messages to train and evaluate the system. For the Facebook dataset, we use the result of one previous study [9], where 199,782 wall posts are confirmed as spam. We further scan through the entire dataset looking for additional wall posts that share either identical textual description or URL with the confirmed spam wall posts. We label the newly found wall posts as spam, too. At last we have 217,802 wall posts labeled as spam. For the Twitter dataset, we visit all the embedded links. In Twitter, all the embedded links sit behind a URL shortening service. We label a URL as malicious if the URL shortening service stops serving the URL due to a policy violation. All Tweets containing the malicious URLs are labeled as spam. In total 467,390 tweets are labeled as spam. Note that in general, we lack comprehensive ground truth. Although all the messages labeled as spam are truly spam, some spam might be missed and labeled as legitimate message. Thus, we may somewhat underestimate false negative rate (spam that the system fails to report) during the evaluation, and overestimate false positive rate (legitimate messages misclassified as spam).



(a) The scenario without the deployment of our system.

(b) The scenario with the deployment of our system.

Figure 1: A simplified OSN architecture, only illustrating the components related to the message exchanging functionality. User A represents the account controlled by spammers.

3 Features

We first briefly review some features used in existing spam detection work but are not suitable for the OSN environment. After that, we investigate various features that can potentially differentiate between spam and legitimate clusters. Note that *not* all features have never been used before. For each feature, we conduct experiment with the labeled Facebook dataset to show their effectiveness. In this section, we cluster the spam and legitimate messages separately to obtain the "pure" clusters, using the approach that is given out in Section 4, which provides the system design detail.

We organize the features into two categories. *OSN specific features* are those that need social network information to compute. *General features* are those that could also be used to detect spam outside OSNs. No single feature is perfectly discriminative between spam and legitimate clusters. Each single feature is likely to fail in some scenario. Instead, these features are used in combination to train the best classifier.

3.1 Obsolete Features

Supervised machine learning has been a popular tool for spam detection. Numerous features have been proposed and work particularly well. However, OSN is a different context where the features' effectiveness needs a reevaluation. Note that we do not review *all* the previously proposed features. Instead, we cover the features used in two state-of-the-art works that detect spam in email [11] and forum [23], believing those features are the most advanced ones. Next, we present the ones found to be problematic.

Message size is a popularly used feature, because legitimate emails have variable sizes while spams tend to be small. We measure the size of the spam and legitimate messages in our dataset, and present the distribution in Figure 2. Apparently both types of messages in OSNs are small. Their distribution heavily overlaps in the area of less than 200 bytes. Consequently, for the majority of messages the size is not a distinguishing feature. Two similar features are the number of words and the average word length in each message. We present their distribution in Figure 3 and Figure 4, respectively. The observation is consistent with the message size feature. Neither of them would remain effective in OSNs.

A series of network based features, which are sender IP neighborhood density, sender AS number and status of sender's service ports, have also been proposed. Although we cannot measure them straightforwardly due to the lack of such information in our dataset, we still find it problematic to use them to detect OSN spam, because the underlying assumption is violated. The assumption is that spam emails are sent out using SMTP servers running on large number of infected ordinary end hosts, whereas legitimate emails originate from a set of dedicated SMTP servers. As a result, the spam senders' IP neighborhood will be crowded due to the botnet infection, but the IP addresses of legitimate SMTP servers are far apart. Also because of the infection pattern of botnet, spamming hosts tend to aggregate into a small number of ASes, which makes sender AS number a promising feature. At last, the spamming bots may not listen to other service ports, but legitimate SMTP servers will do so. Unfortunately, in OSNs both spam and legitimate messages are sent from ordinary end hosts. Given the large number of OSN users, the sender's IP neighborhood is expected to be dense for both spam and legitimate messages. In addition, there would be no difference in the status of sender's service ports for the two cases. At last, spam senders may still be aggregated into a small number of ASes. However, legitimate message senders will be mixed

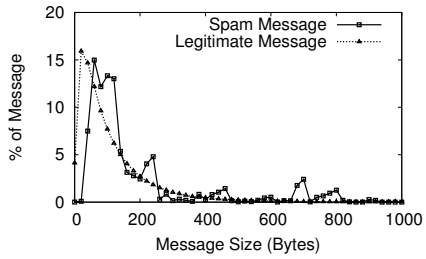


Figure 2: Distribution of spam and legitimate message size, respectively. Each bin is 20 bytes.

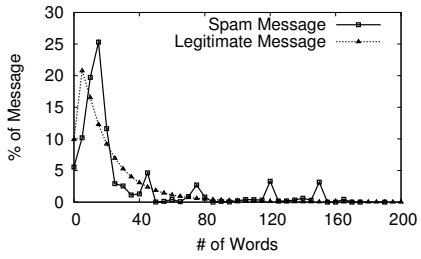


Figure 3: Distribution of spam and legitimate message word count, respectively. Each bin is 5 words.

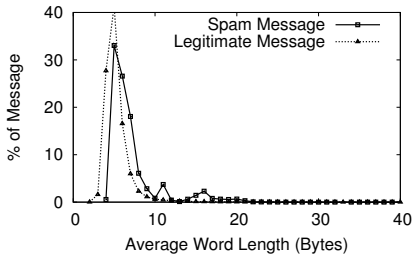


Figure 4: Distribution of average word length of spam and legitimate messages, respectively. Each bin is 1 byte.

into the same ASes and the effectiveness of sender AS number as a feature would be seriously degraded.

3.2 OSN Specific Features

OSN users form a huge social graph, where each node represents an individual user. In Facebook-like OSNs, a social link would connect two nodes if the two corresponding users have mutually agreed to establish a social connection. Two users without a social link between them cannot directly interact with each other. Twitter-like OSNs impose looser restrictions, where a user can "follow" anyone to establish a directed social link, so that he can receive all the updates. Similarly, the interaction history among users forms an interaction graph.

Sender Social Degree Recent studies suggest that the majority of spamming accounts in OSNs are compromised accounts [9, 10]. Meanwhile, research on modeling epidemics in topological networks indicates that the more edges a node has, with a higher probability it will be infected quickly by an epidemic [6, 36]. On the other hand, the accounts created by spammers also have an incentive to make large number of friends to open up communication channels. Consequently, we hypothesize that spamming accounts have higher degree in the social graph than legitimate accounts. In Twitter-like OSNs, we define the social degree of one node as the number of other nodes it "follows", since this number is controlled by the account itself. In Facebook-like OSNs, there is no ambiguity as edges are not directed.

Figure 5 shows that our intuition is roughly correct. The average social degree of spamming accounts is 59.2. It is about 50% higher than that of legitimate accounts, which is 40.8. Given such an observation, it is tempting to use the message sender's social degree as one feature. Unfortunately, the compromised spamming account will send a mixture of spam and legitimate messages. Hence in many cases spam and legitimate messages share the same sender and will naturally have exactly the same value of the sender's social degree. However, after being organized into

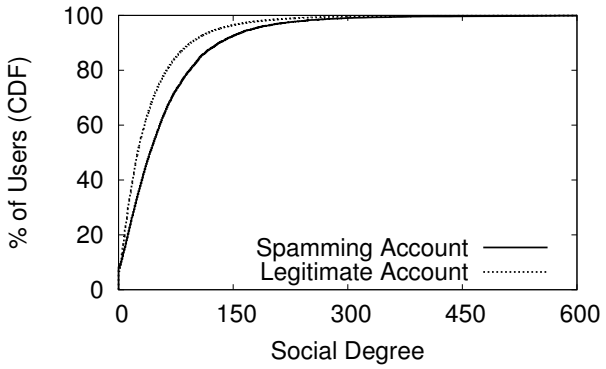


Figure 5: Cumulative distribution of the social degree of spamming and legitimate accounts, respectively.

clusters, the average senders' social degree of the clusters becomes an effective feature. The main reason is that *all* senders in spam clusters are spamming accounts, while this is very unlikely to be true for legitimate clusters. As a result, spam clusters are expected to exhibit higher average senders' social degree. Figure 6 confirms this expectation. It does not depict the clusters whose senders' social degree is not available in the dataset. Despite some overlapping in the lower left region, the solid curve representing spam clusters is to the right of the dashed curve representing legitimate clusters. About 80% of legitimate clusters have an average senders' social degree of less than 400. In contrast, this value is larger than 400 for about 50% of spam clusters.

Interaction History Although one account may establish a large number of social links in the social graph, it only interacts with a small subset of its friends [29]. However, its behavior is expected to deviate from this pattern once the account is compromised and under the control of spammers, since the spammers desire to push spam messages to as many recipients as possible. Consequently, a sudden burst that the accounts start to interact with the friends that they do not, or rarely, interact with before becomes a strong signal indicating spamming activities. Note that this intuition

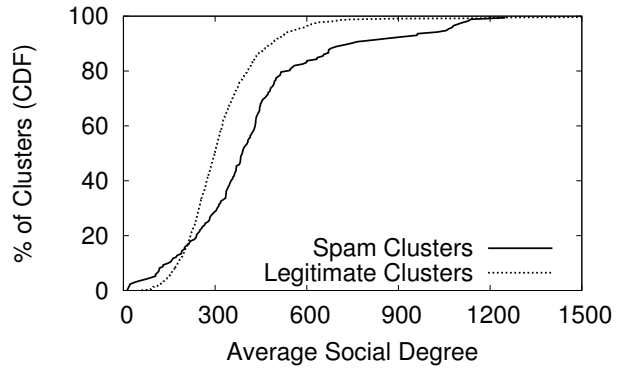


Figure 6: Cumulative distribution of average senders' social degree of spam and legitimate clusters, respectively.

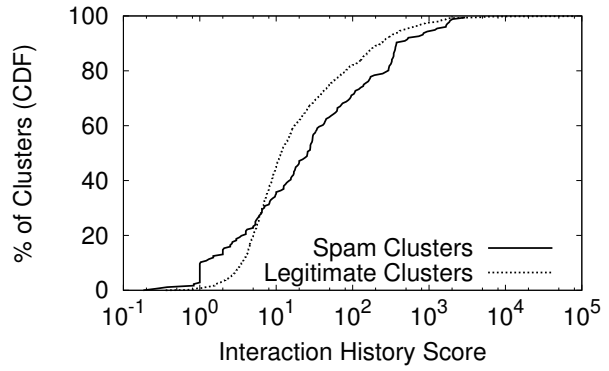


Figure 7: Cumulative distribution of interaction history score of spam and legitimate clusters, respectively.

applies specifically to Facebook-like OSNs, since messages are always broadcast to all followers in Twitter-like OSNs.

We use the notion of "interaction history score" to quantify this intuition. The k_{th} interaction between a user pair is weighted $1/k$, so that messages sent between user pairs that rarely interact are given heavier weight values. As k increases, the message weight between frequently interacting user pairs decreases. We define the interaction history score of a cluster as the sum of the message weight within the cluster. Naturally, the spam clusters are expected to have higher score because larger number of less frequently interacting user pairs is contained. Figure 7 plots the CDFs of this score for spam and legitimate clusters, respectively. For the 20% clusters with the lowest scores, the expectation is not always true. However, spam clusters indeed exhibit higher score for the rest of the cases. Note that the x-axis is in log scale. The score of spam clusters is usually several times higher.

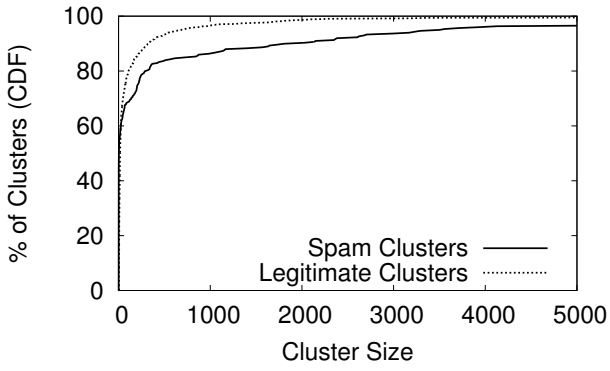


Figure 8: Cumulative distribution of spam and legitimate cluster size, respectively.

3.3 General Features

We find four additional features effective to discriminate between spam and legitimate clusters. These features do not need the social graph or the interaction graph to compute, and could also apply to spam detection problems outside OSNs. They are denoted as the general features.

Cluster Size Although each spamming account might not generate large number of spam messages, the spam campaign as a whole must contain large number of spam messages in order to be prosperous. Figure 8 plots the CDFs of the size, measured as the number of message contained, for spam and legitimate clusters, respectively. We observe that about 50% of spam clusters' size is less than 10, which causes the overlapping part of the two curves. On the other hand, large spam clusters exhibit a big difference in size comparing with legitimate clusters. The overlapping of size of small clusters does not make this feature invalid. The reason is that small spam clusters only have minor impact on the system's detection performance. Instead, the big clusters are those that matter. As an extreme example, it is still acceptable even if the system correctly identifies 10% of spam clusters of size 1,000 while missing the remaining 90% of spam clusters of size 10.

Average Time Interval Known as the "bursty" property, most spam campaigns involve coordinated action by many accounts within short periods of time [30]. The effect is that messages from the same campaign are densely populated in the time period when the campaign is active. Consequently, if we compute the intervals between the generation time of consecutive messages in each cluster, the spam clusters are expected have shorter intervals than the legitimate clusters. From a statistical prospective, the median interval would nicely quantify the "bursty" property, and is robust to outlier values. However, it is expensive to maintain during runtime, as the complete list of intervals must be recorded and

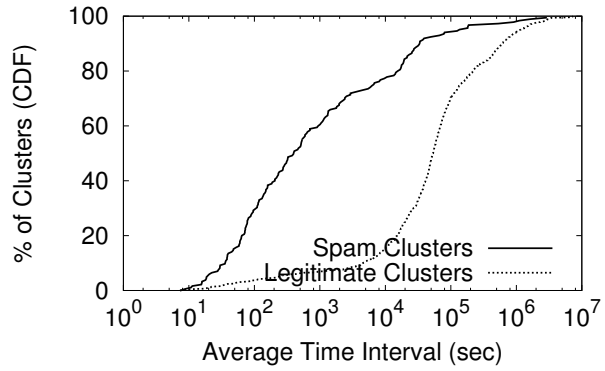


Figure 9: Cumulative distribution of average time interval of spam and legitimate clusters, respectively.

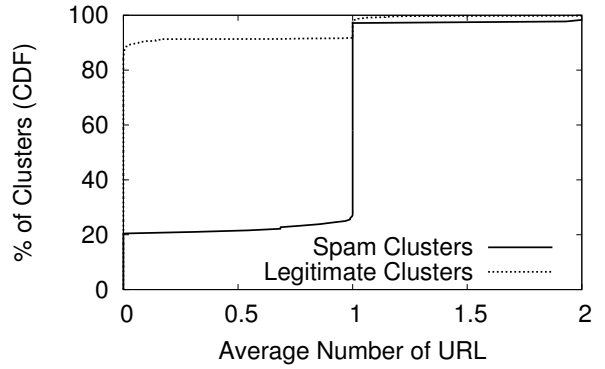


Figure 10: Cumulative distribution of average number of URLs per message of spam and legitimate clusters, respectively.

sorted. Alternatively, we quantify this property using the average time interval, which is much more "lightweight", since only the starting time, ending time and the total number of messages in the cluster are needed to calculate its value. Figure 9 plots the CDFs of average time interval of spam and legitimate clusters, respectively. The x axis is in log scale so that the figure is readable. It shows that the average time interval of legitimate clusters is orders of magnitude larger than that of spam clusters.

Average URL Number per Message Spammers always solicit some action from the recipients, *e.g.* , to click a link, in order for economic benefit ultimately. As the most commonly adopted practice, spam messages are embedded with URLs that the spammers want to advertise. In contrast, legitimate messages do not contain URLs in most of the case. Hence, we use the average number of URLs per message in each cluster as one feature. Figure 10 plots the CDFs of this feature for spam and legitimate clusters, respectively. It shows that more than 80% of benign clusters do not contain any URL. Only about 2% benign clusters contain more than

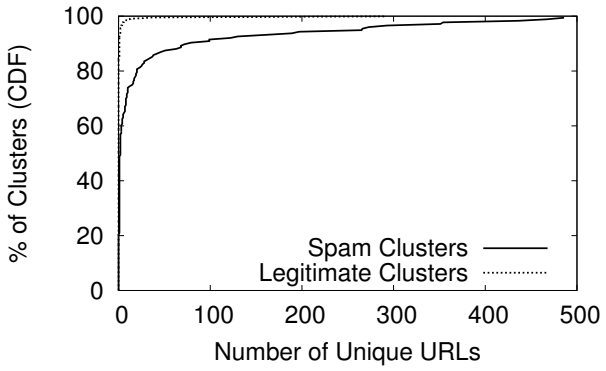


Figure 11: Cumulative distribution of number of unique URLs contained in spam and legitimate clusters, respectively.

one URL in each message on average. In comparison, about 80% of malicious clusters contain at least one URL in each message on average.

Unique URL Number URL blacklist has been extensively used nowadays. As an attempt to evade blacklist based detection, spammers have evolved to embed different URLs in messages from the same campaign, with each URL being used less frequently, so that some, if not all, of the spam URLs will not be blacklisted. This type of spam campaigns will inevitably contain large number of unique URLs. On the other hand, legitimate clusters are not likely to exhibit the same behavior. Figure 11 plots the CDFs of the number of unique URLs contained in spam and legitimate clusters, respectively. The curve representing legitimate clusters embraces the y axis, which is consistent with the fact that many legitimate messages do not contain URL. On the contrary, about a half of the spam clusters have multiple URLs. The most diverse cluster uses 486 different URLs.

4 System Design

In this section, we present the detailed design of the online spam filtering system. Section 4.1 first introduces the system overview. After that, Section 4.2 and Section 4.3 present the incremental clustering module and the supervised machine learning module, respectively. Finally, Section 4.4 illustrates an extension on the basic design that exploits parallelism to boost the system throughput.

4.1 Overview

Figure 12 shows the system design overview. The intuition is that spam campaigns are generated using templates and that messages in the same campaign shall retain certain similarity among them. After a clustering process, spam

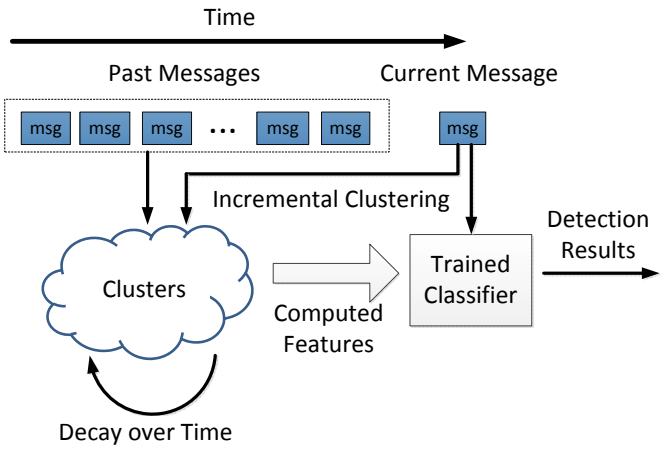


Figure 12: The system design overview.

messages in the same campaign will reside in the same cluster or only a small number of clusters. Note that we do not depend on the system to agglomerate an entire campaign into one single cluster, which is impractical due to the variation brought by the spam template. However, because the total number of messages in a spam campaign is very large, each spam cluster will still contain significant number of messages. In addition, the clusters formed by the spam messages will exhibit different characteristics comparing to the legitimate clusters as Section 3 shows. Hence, they can be differentiated using supervised machine learning.

Accordingly, the two major components in the system are the *incremental clustering module* and the *supervised machine learning module*. The first module maintains a collection of disjoint clusters such that each message that has been processed in the past is present. The second module is essentially a trained classifier that makes binary decisions. When the system receives a new message, it first incrementally updates the clustering result with minimal computational overhead. The new message may form a new cluster by itself, be incorporated into one existing cluster or even trigger the merge of multiple existing clusters. In the first case, the system directly marks the message as "legitimate" without invoking the classifier, because the average time interval feature cannot be calculated with only 1 message. This implicit limitation will cause a false negative only when the new message is the first one in a spam cluster, which happens very rarely. In any other case, the values of the six features of the cluster that the new message resides in are (re-)calculated. The other clusters are intact. After that, the trained classifier accepts these values and decides whether these values represent a spam cluster. Note that if the classifier outputs "spam", it will only trigger a spam alarm on the current message, rather than all the messages in the cluster. Since it is an online system, the decision on the previous messages in the same cluster has already been made and is not changed.

A practical concern is that the hardware resource is always limited. It is not affordable to keep track of all the messages observed in the past. In addition, recent messages should be given heavier weight that influences the clusters' feature values. To address these two concerns, the clusters that the system maintains decays exponentially over time. More specifically, once after the system has processed w messages, it shrinks the six feature values associated with each cluster by a factor of a, where w and a are two important system parameters to determine the decaying speed. If a cluster's size is shrunk to a value below a threshold t, it is permanently removed and all the resources it takes up are freed.

4.2 Incremental Clustering

The ability to do incremental clustering is the key to the system. The first design choice we must make is to define the distance between two messages. Naturally, using semantic similarity will result in more accurate clustering result as messages in the same campaign are more likely to have small distance. However, semantic analysis involves heavier computational overhead that is not acceptable for a real-time system. Alternatively, we choose text shingling [7] as the basic tool to calculate the distance between messages. Text shingling is initially proposed to analyze web page similarity in huge volume and is sufficiently lightweight. In the shingling process, the original message is disassembled into fixed-length substrings, each of which is called a "shingle". We sort the hashed shingle and use the first 20 of them to represent each message. For the messages with less than 20 shingles, the system does not cluster them and directly marks them as legitimate if they do not contain any URL, since they are too short to carry out attacks. The similarity between a message pair is estimated by its *resemblance* score, defined as the ratio of shared shingles to all unique shingles. Two messages are deemed as "similar" if their resemblance score surpasses a predefined threshold.

We augment the original text shingling with URL comparison as an adaptation to the specific application of spam detection. URLs embedded in the spam messages represent the spammers' goal. Hence two spam messages sharing the same embedded URL shall come from the same campaign, although their textual description might be quite different judged by text shingling due to the usage of templates. To sum up, two messages are considered as "similar" if their resemblance score is sufficiently large *or* their embedded URLs are identical. They will reside in the same cluster if either condition is satisfied. A similar method was also used by Zhuang *et al.* [35] to analyze email spam traces offline.

We use a hash table to store the text shingling result, indexed by the shingles. It keeps track of all the messages that contain the corresponding shingle. Meanwhile, we use another hash table to store the result of URL comparison. It is indexed by the URLs and each element is a list of messages that contain the corresponding URL. When a new message arrives, it is disassembled into shingles and all the previously observed messages with identical shingles can be conveniently identified. The same holds for the URL comparison. In this way, the system does not need to compare the new message with all observed messages sequentially. Instead, it directly looks up the hash tables at the location of matching shingles and URLs, thus minimizing the overhead.

For the purpose of incremental clustering, our system maintains a set of disjoint clusters. Upon the arrival of a new message, it first creates a cluster containing a single element, the new message itself. Next, all clusters that contain a message that is "similar" to the new message must be found and merged with the new cluster, while the other clusters stay intact. We build a Union-Find data structure to accommodate this need. A Union-Find data structure supports three operations very efficiently, which are i) creating a new set, ii) finding the set containing a given element and iii) merging two sets. In addition, the system needs to eliminate clusters from time to time because of the exponential decaying. This operation is not supported by the standard Union-Find data structure. As a result, we enhance it to provide the functionality to return all the elements of a given cluster.

4.3 Supervised Machine Learning

While clustering provides a division of the observed messages, our system needs to make decision on each incoming message that flags it as either a spam or a legitimate message. The supervised machine learning module is essentially a trained classifier that makes this binary decision.

Two classifier candidates, support vector machine (SVM) [8] and decision tree [21], are widely used in the literature. Decision tree has the advantage that the trained classifier is simple to understand and fast. The time complexity to make a prediction on a new test point is $O(log(N))$, where N is the number of nodes in the trained tree. Such efficiency is a desirable property for an online system. On the other hand, SVM is reported to achieve good accuracy on a wide variety of problems such as handwriting recognition, face detection, text categorization, *etc.* However, the classification process of new data points gets slower if the training set is large. We test both classifiers on our dataset and decision tree yields better accuracy, *i.e.* , higher true positive rate and lower false positive rate. Consequently, we pick decision tree as the classifying module in the system.

4.4 Parallelization

Our system needs to achieve high throughput as an online spam filtering tool. The basic design shown in Figure 12 spends the majority of running time on incremental clustering. Consequently, accelerating the clustering process can greatly enhance the throughput.

Parallelization is a natural choice to achieve such a goal, but it seems to be counterintuitive in our specific case at the first glance, since incremental clustering is a sequential process, in the sense that the clustering process of the $n + 1_{th}$ message is based on the clustering result of the first n_{th} messages. Our solution is to compute the incremental clustering result for the next k messages, instead of the next one message, simultaneously. For each one of the next k messages, we compare its similarity to the observed messages and decide which existing clusters it should be merged into. The computation for one message is independent from others. Accordingly, k threads are executed at the same time with each thread handling one message. Note that only the message similarity comparison is parallelized. All other operations, such as cluster merging, remain sequential to avoid conflict among threads.

The parallel design might slightly alter the clustering result comparing to the sequential case, but it will not noticeably affect the detection result. Without loss of generality, we analyze the potential alteration under the scenario of two threads, t_1 and t_2, processing two messages, m_1 and m_2, in parallel. The only situation when the clustering result differs must satisfy two conditions: i) m_1 and m_2 are similar, ii) there does *not* exist a previously observed message m_0, such that m_0 is similar to both m_1 and m_2. In this situation, a sequential execution will cluster m_1 and m_2 together immediately, but a parallel execution will not do so. The reason is that t_1 is unaware of m_2 and cannot find any previously observed message that is similar to m_1. Hence m_1 is left in a separate cluster containing only itself. The same applies to m_2. However, their corresponding clusters will be merged once a third message, m_3, which is similar to both m_1 and m_2, has been processed. In the context of spam message detection, the above scenario maps to the case that the first two spam messages from the same campaign are handled by the system in parallel. Given the limited number of spam campaigns, this scenario shall happen very rarely. For legitimate messages, such inconsistency in the clustering process will not affect the detection result at all since any clusters with size of 1 will be assumed to be legitimate.

5 Experiment

5.1 Methodology

We evaluate the system on a server machine that has eight cores (Xeon E5520 2.2Ghz) with Hyper-Threading and 16GB memory. We sort all the messages according to their timestamps and divide them into the training set and the testing set. The training set includes the first 25% of spam messages and all the legitimate messages in the same time period. The testing set contains the remaining messages.

The system has multiple tunable parameters (Section 4.1). W and a control the speed of cluster decay. The choice of their value is mainly constrained by the hardware resource, as lower decaying speed results in more resource consumption and slower processing speed. We pick 100,000 and 0.2 as the value of w and a in the experiments, respectively, according to our hardware resource limit. We also tested faster decaying speed by setting a as 0.5, but only found very slight change in the detection result. We pick 3 as the value of t, the size threshold to eliminate a cluster, since a small value reduces the risk of removing a live spam cluster from the system.

In the training phase, we first feed the incremental clustering module with the training set, and record the feature values for all spam and legitimate clusters. We use all 6 features for the Facebook dataset. We use the 5 features other than interaction history score for the Twitter dataset, since interaction history score is not applicable for the broadcast interaction in Twitter. Next, We use the extracted feature values to train the classifier. We only use the clusters whose size is at least 5 for training. After that, the system is fed with the testing set. We conduct all the experiments obeying the time order, so that the experimental results can resemble the system performance in real-world deployment accurately.

5.2 Clustering Resemblance Threshold

The detection is based on the clustering of spam messages. Hence, the resemblance threshold used in the clustering process could potentially affect the detection result. Recall that the resemblance of a message pair is the ratio of shared shingles to the total number of distinct shingles, which represents the "similarity" between the message pair. In order to understand how to pick the best threshold, we study the resemblance value among spam and legitimate message pairs, respectively.

We first divide the Facebook dataset into the spam set and the legitimate message set. The division is trivial using the label of each message. After that, we compute the resemblance value of all the spam message pairs and the legitimate message pairs, respectively, by repeating the clustering process. Figure 13 shows the cumulative distribution of such value. The curve for legitimate message pairs embraces the y axis. For both types of message pairs, most of them have a very small resemblance value, meaning that they are almost totally dissimilar. This is expected because legitimate messages are spontaneously generated and are different to others naturally. For spam messages, many campaigns exist in our dataset and the message pairs across different campaigns are also very different. However, the curve for spam message pairs exhibits a sudden rise at the right most part, showing that some message pairs are very similar, which are those belonging to the same spam campaigns. The legitimate message pairs do not exhibit such a pattern due to the spontaneous nature. Despite this difference, it is apparent that both curves are flat in the middle. It suggests that comparably very few message pairs have the resemblance value falling in the middle area. Consequently, the system is insensitive to the threshold value as long as the value is not too small or too large, since varying the threshold would not significantly affect the clustering result. At last we pick 0.5 as the threshold used in all the experiments.

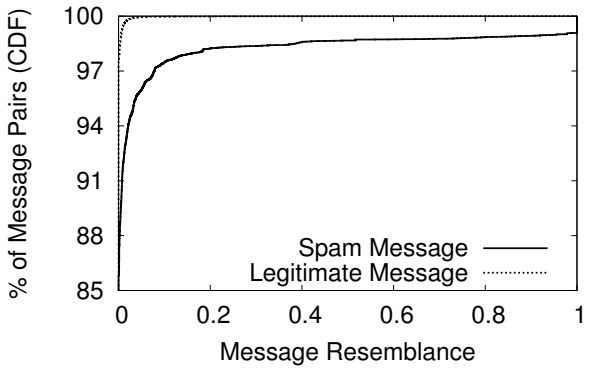


Figure 13: The distribution of resemblance value among spam and legitimate message pairs in the Facebook dataset, respectively.

5.3 Accuracy

The accuracy of a detection system is characterized by two metrics, *true positive rate* (TPR) and *false positive rate* (FPR). True positive rate shows the detection completeness. It is defined as the number of instances correctly classified as spam divided by the total number of spam instances. False positive rate reflects the detection error on the legitimate messages. It is defined as the number of instances incorrectly classified as spam divided by the total number of legitimate instances. In this section we evaluate the overall accuracy (Section 5.3.1), the accuracy of different fea-

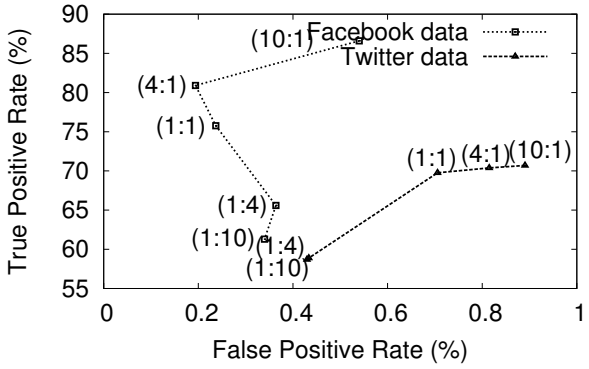


Figure 14: The overall detection accuracy subjecting to the varying spam to legitimate message ratio in the training set.

ture sets (Section 5.3.2) and the accuracy over time (Section 5.3.3) to show that our system can indeed provide satisfactory accuracy guarantee.

We use the C4.5 decision tree library provided by Quinlan [20].

5.3.1 Overall Accuracy

We first test the overall detection accuracy, using all applicable features and the complete dataset. An accurate detection system desires high true positive rate and low false positive rate simultaneously. Unfortunately, parameter tuning usually causes these two metrics to increase or decrease at the same time. We are forced to face the trade-off between them. We decided to tune the classifier to emphasize low false positive rate while maintaining a reasonably high true positive rate.

As suggested by Zadrozny *et al.* [33] and used by Thomas *et al.* [27], we adjust the ratio of spam to legitimate message in the training set by random sampling to tailor the performance, where a larger ratio indicates a stronger tendency to classify a message as spam. We use the ratios of 10:1, 4:1, 1:1, 1:4 and 1:10. Regardless of the ratio in the training set, the full testing set is used. Figure 14 shows the detection accuracy. For the Twitter dataset, the two rates grow in consistency with the ratio in the training set. Both rates minimize when the ratio is 1:10, and maximize when the ratio is 10:1. A 1:1 ratio in the training set results in a reduction of false positive rate by about 20%, comparing to a 10:1 ratio. On the other hand, the true positive rate decreases very slightly and reaches 69.8% at this ratio. A 1:4 ratio or a 1:10 ratio results in a substantial drop in both true positive rate and false positive rate. After all, we still favor the 1:1 ratio for the twitter data. The detection accuracy of the Facebook dataset exhibits irregularity. A 4:1 ratio yields the lowest false positive rate (0.19%) and a relatively high true positive rate (80.9%). For the remainder of the evaluation, we conduct all experiments over the Facebook dataset at a 4:1 ratio in the training set and all experiments over the Twitter dataset at a 1:1 ratio.

Feature Set	Features Contained	TPR	FPR
OSN Specific	Social Degree Interaction History	38.3%	0.30%
General	Cluster Size Average Time Interval Average URL Number Unique URL Number	80.8%	0.32%

Table 1: The detection accuracy using each of the two feature sets with the Facebook dataset.

5.3.2 Accuracy of Different Feature Sets

As stated in Section 3, we divide our features into two sets: the OSN specific features and the general features. To understand their significance to the system, we train the classifier exclusively using each feature set, test the detection accuracy and present the result in Table 1. The ratio of spam to legitimate message in the training set is 4:1. Since the Facebook dataset uses the complete 6 features, we conduct this experiment using the Facebook dataset. The general features achieve a true positive rate of 80.8%, which is similar to the result of using all features. Unfortunately, the false positive rate increases by more than 50%. On the other hand, the OSN specific features lead to a lower true positive rate (38.3%). We do not mean to compare between these two sets and decide which one gives better performance. Rather, the result shows that the detection accuracy would be significantly degraded in the absence of either feature set. Using their combination keeps the high true positive rate while reducing the false positive rate.

5.3.3 Accuracy Over Time

Spam campaigns are not persistent. Criminals will promote new campaigns on a continuous basis. As a result, it is important to evaluate how much time it takes before the classifier becomes out of date and needs to be re-trained. We carry out the evaluation using the Facebook dataset, because it contains messages generated throughout a time period of one year and a half. In comparison, the Twitter dataset is collected in a much shorter period of time and is not suitable for this type of evaluation. We use 4:1 as the ratio of spam to legitimate message in the training set. We dissect the testing set into 3-month time periods and study how the detection accuracy changes over time. We use the same trained classifier on the first 9 months of testing data, and measure the true positive rate as well as the false positive rate in each period. We only present the result in the first

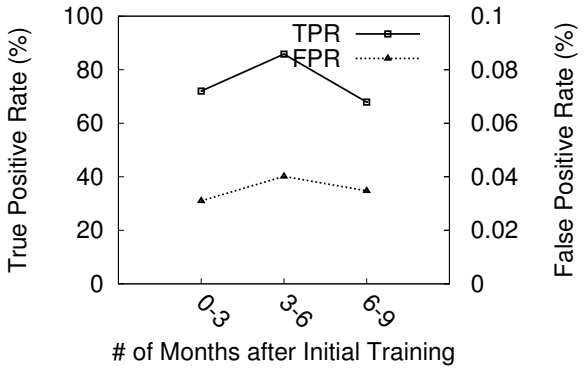


Figure 15: The detection accuracy on the Facebook dataset over time.

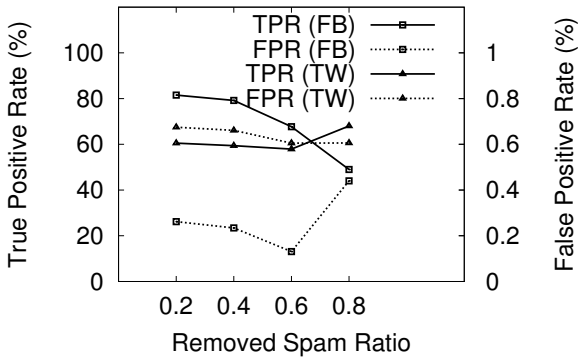


Figure 16: The detection accuracy on the Twitter dataset under stealthy attack.

9 months, because after 9 months the spam becomes very sparse in our dataset and the true positive rate is not representative. However, we do observe a significant rise in the false positive rate after 9 months, suggesting that our system may need re-training after such a long time to further reduce the false positive rate. Figure 15 shows the experimental result.

The false positive rate remains extremely low for all 3 periods. The period between 3 and 6 months after training incurs the highest false positive rate, which is about 0.04%. It shows that our system misclassifies very few legitimate messages as spam, even if the training takes place up to 9 months before the time of classification. The true positive rate in the first period and the third is a bit lower. We find that it is caused by campaigns heavily adopting obfuscation that results in large number of small clusters instead of small number of big ones. This type of campaign does not appear in the second period. It demonstrates that the features we select indeed capture the commonality among spam campaigns, so that they remains effective even if the campaigns being classified do not present in the training data. These findings indicate that our system can remain accurate without the need for re-training for a long period of time, which minimizes the maintenance cost after deployment.

5.4 Resilience against Stealthy Attack

One of the spammers' common attempts to evade detection is to carry out stealthy attack. Under stealthy attack, the speed of spam generation is reduced so that the anti-spam system cannot acquire sufficient instances to learn the spam pattern. To simulate such an attack, we randomly remove certain proportion of spam messages , ranging from 20% to 80%, from both dataset. The effect is the same as if the spammers generate spam messages at a reduced speed. The legitimate messages are untouched. After that, we repeat the overall accuracy test on the modified dataset, choosing 4:1 and 1:1 as the ratio of spam to legitimate messages in the training set for Facebook and Twitter dataset , respectively. Figure 16 illustrates the result. In the Twitter experiment, the true positive rate and the false positive rate exhibit very slight difference comparing with the experiment using the original dataset in all test cases. In the Facebook experiment, the true positive rate and the false positive rate decrease moderately when we remove 20%, 40% and 60% of spam messages. The reason is that as spam becomes more sparse in the dataset, more spam clusters become indistinguishable from legitimate message clusters. When we remove 80% of spam messages, the system does not handle it gracefully as the true positive rate drops to 49% and the false positive rate rises to 0.44%. The results show that except for the Facebook experiment when we remove 80% of spam messages, stealthy attack slightly decreases the system's the true positive rate. Nonetheless, the false positive rate does not increase, which means that the system is still "safe" against legitimate messages under such attack.

5.5 Run Time Performance

The run time performance is critical, since our system must not be the bottleneck that slows down the OSN platform if deployed. The training time is only a few seconds in all the experiments. In addition, we measure the latency and throughput to demonstrate that our system can respond to an incoming message very quickly and that it can inspect large number of messages per second.

Latency The latency is measured as the time between when the system receives a message and when the system outputs the inspection result. We process all the messages sequentially in this experiment. Figure 17 shows the cumulative distribution of the system latency, measured in milliseconds. The average and median latency for the Face-

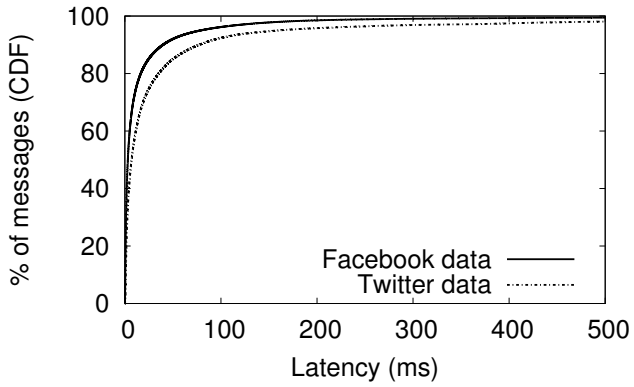


Figure 17: The cumulative distribution of the system latency, measured in milliseconds.

book dataset is 21.5ms and 3.1ms, respectively. The average and median latency for the Twitter dataset is 42.6ms and 7.0ms, respectively. Even for the Twitter dataset that results in comparatively longer processing delay, the whole inspection process for over 90% of messages is completed within 100ms. Given that modern browsers spend several hundred milliseconds before they start to render ordinary webpages [17], the system latency is sufficiently low. An OSN user would not feel any noticeable additional delay due to the deployment of our system.

Throughput In order to test the throughput, we feed the system with the testing set as fast as possible. Due to the hardware limitation (8 physical cores with Hyper-Threading), we execute 16 threads simultaneously. We divide the total number of messages processed by the system by the total running time to calculate the average throughput. The throughput on the Facebook and the Twitter dataset is 1580 messages/sec and 464 messages/sec, respectively.

6 Discussion

Spammers continuously rival anti-spam solutions. Although our system is demonstrated to be robust against stealthy attacks, other approaches might be adopted to evade it. One possible way is to tamper with the clustering process by reducing the syntactic similarity among messages from the same campaign. We have observed such attempts in both of our datasets. In the Facebook dataset, spam campaigns start to include obfuscation in the textual message, *e.g.* , using "profil imaage" instead of "profile image". In the Twitter dataset, almost all spam tweets contain random meaningless words, which we suspect to be obtained from a collection of popular words. Nonetheless, our system defeats these attempts and yields good detection accuracy. Meanwhile, obfuscation and embedded random chunks decrease the message readability as well as trigger the recipients' suspicion, which lowers the underlying conversion rate. Another way to evade the system would be to manipulate the features of spam clusters, for the purpose of making them indistinguishable from legitimate clusters. It is not too difficult to do this for each individual feature. For example, spammers may use fake accounts with lower social degree to tamper with the "sender social degree" feature. In addition, they may only send spam to the friends that the compromised accounts frequently interact with to tamper with the "interaction history" feature. They may also send spam with longer time interval to tamper with the "average time interval" feature. However, manipulating any individual feature is likely to have very limited effect, since our system uses all six features in combination. Even though deliberate spammers in theory may manipulate all the features simultaneously, they have to pay a heavy price for doing so. In the previous three examples, the spam recipient population becomes much smaller, and the speed of spam generation becomes slower by orders of magnitude. If we can force spammers to do that, it is already a victory. The third evasion technique is to produce image-based spam. Our current design does not work for image-based spam.

7 Related Work

We discuss prior related work by organizing them into two general categories: studies of spamming problem in OSNs and in other environments.

Spam Studies in OSNs. Stein *et al.* present the framework of the adversarial learning system that performs real-time classification on read and write action in Facebook [25]. However, the lack of details, *e.g.* , the features and policies used, and performance results prevent us from further comparison. Two offline studies have revealed large-scale spam campaigns in Twitter and Facebook, respectively [9, 10]. Designed as offline analysis tools, none of them can be directly used for online spam detection. [10] is based on URL blacklists which have too long lag-time to protect a significant number of users. [9] uses a similar clustering technique. However, it needs to operate on the complete set of messages, and its efficiency limits the clustering to be performed on the 2 million messages with URLs. In comparison, we adopt incremental clustering and parallelization to enable the system to inspect messages only based on other messages observed in the past, as well as to increase the scalability. Also, we develop the feature set to distinguish spam cluster that can be efficiently computed online. Thomas *et al.* [27] propose to filter malicious URLs in OSNs in real-time, which can be used to identify malicious messages later. While their approach does deep analysis of the URLs' landing pages, our approach uses an

alternative information source for the investigation, *i.e.* , the message content. Song *et al.* propose to use sender-receiver relationship to classify twitter messages [24]. Stringhini *et al.* [26] , Lee *et al.* [14] and Yang *et al.* [31] use machine learning techniques to detect spamming bots in OSNs, which are accounts created by spammers and used exclusively for spamming. Yardi *et al.* use a collection of ad-hoc criteria to identify spamming bots on Twitter [32]. In comparison, our detection focuses on spam messages instead of accounts, so that we can secure OSNs from spams generated by both spamming bots and previously legitimate accounts that have been compromised. Additionally, Benevenuto *et al.* [5] and Markines *et al.* [16] apply supervised machine learning to detect spammers in Youtube and social bookmarking sites, respectively. These websites do not focus on communications among users. Instead, their key functionality is video and bookmark sharing. Hence, the feature set used for machine learning is very different.

Other Spam Studies. There is a large body of prior work studying the characteristics of email spam [4, 12, 13, 35]. However, few of them can potentially be used as online detection tools. Researchers propose to enhance IP blacklists by correlating IP addresses according to their email sending history, network-level information, blacklisting history and so on, in order to discover spamming IP addresses that are not yet blacklisted [19, 22, 28]. As discussed previously, sender reputation based approaches are not suitable for OSN spam detection. Pitsillidis *et al.* extract the underlying templates to match future spams [18]. Li *et al.* propose to enhance Bayesian filter by adjusting the weight of spam keywords using personal social network information [15]. Both approaches detect campaigns that have been contained in the training set. Our approach does not have such a restriction. Xie *et al.* generate regular expression signatures for spamming URLs [30]. In addition, Thomas *et al.* use a comprehensive list of features to determine whether a given URL directs to spam in real-time [27]. The above two approaches essentially detect spamming URLs. In comparison, our approach can detect spam messages even if no URL can be recognized in them due to the obfuscation techniques.

8 Conclusions

In this paper, we describe our work to provide online spam filtering for social networks. We use text shingling and URL comparison to incrementally reconstruct spam messages into campaigns, which are then identified by a trained classifier. We evaluate the system on two large datasets composed of over 187 million Facebook wall messages and 17 million tweets, respectively. The experimental results demonstrate that the system achieves high accuracy, low latency and high throughput, which are the crucial properties required for an online system. In addition, the system is able to remain accurate for more than 9 months after the training phase, which shows its very low maintenance cost after deployment. For more information, please refer to the project web page at http://list.cs.northwestern.edu/.

Acknowledgments

We express our sincere thanks to the anonymous reviewers for their valuable feedback. The authors denoted with † are supported by NSF award numbers CCF-0621443, OCI-0724599, CCF-0833131, CNS-0830927, IIS-0905205, OCI-0956311, CCF-0938000, CCF-1043085, CCF-1029166 , and OCI-1144061.

References

[1] Users of social networking websites face malware and phishing attacks. Symantec.com Blog.

[2] What the trend. `http://www.whatthetrend.com/`.

[3] Zeus botnet targets facebook. `http://blog.appriver.com/2009/10/zeus-botnet-targets-facebook.html`.

[4] ANDERSON, D. S., FLEIZACH, C., SAVAGE, S., AND VOELKER, G. M. Spamscatter: characterizing internet scam hosting infrastructure. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (Berkeley, CA, USA, 2007), USENIX Association, pp. 10:1–10:14.

[5] BENEVENUTO, F., RODRIGUES, T., AND ALMEIDA, V. Detecting spammers and content promoters in online video social networks. In *Proc. of SIGIR* (Boston, Massachusetts, USA, July 2009).

[6] BOGUÑÁ, M., PASTOR-SATORRAS, R., AND VESPIGNANI, A. Epidemic spreading in complex networks with degree correlations.

[7] BRODER, A. Z., GLASSMAN, S. C., MANASSE, M. S., AND ZWEIG, G. Syntactic clustering of the web. *Comput. Netw. ISDN Syst. 29* (September 1997), 1157–1166.

[8] BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov. 2* (June 1998), 121–167.

[9] GAO, H., HU, J., WILSON, C., LI, Z., CHEN, Y., AND ZHAO, B. Y. Detecting and characterizing social

spam campaigns. In *Proceedings of the 10th annual conference on Internet measurement* (New York, NY, USA, 2010), IMC '10, ACM, pp. 35–47.

[10] GRIER, C., THOMAS, K., PAXSON, V., AND ZHANG, M. @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 27–37.

[11] HAO, S., SYED, N. A., FEAMSTER, N., GRAY, A. G., AND KRASSER, S. Detecting spammers with snare: spatio-temporal network-level automatic reputation engine. In *Proceedings of the 18th conference on USENIX security symposium* (Berkeley, CA, USA, 2009), SSYM'09, USENIX Association, pp. 101–118.

[12] KANICH, C., KREIBICH, C., LEVCHENKO, K., ENRIGHT, B., VOELKER, G. M., PAXSON, V., AND SAVAGE, S. Spamalytics: An empirical analysis of spam marketing conversion. In *Proc. of the ACM Conference on Computer and Communications Security* (October 2008).

[13] KREIBICH, C., KANICH, C., LEVCHENKO, K., ENRIGHT, B., VOELKER, G., PAXSON, V., AND SAVAGE, S. Spamcraft: An inside look at spam campaign orchestration. In *Proc. of LEET* (2009).

[14] LEE, K., CAVERLEE, J., AND WEBB, S. Uncovering social spammers: social honeypots + machine learning. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2010), SIGIR '10, ACM, pp. 435–442.

[15] LI, Z., AND SHEN, H. SOAP: A Social Network Aided Personalized and Effective Spam Filter to Clean Your E-mail Box. In *Proceedings of the IEEE INFOCOM* (April 2011).

[16] MARKINES, B., CATTUTO, C., AND MENCZER, F. Social spam detection. In *Proc. of AIRWeb* (2009).

[17] MOSHCHUK, A., BRAGIN, T., DEVILLE, D., GRIBBLE, S. D., AND LEVY, H. M. Spyproxy: execution-based detection of malicious web content. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (Berkeley, CA, USA, 2007), USENIX Association, pp. 3:1–3:16.

[18] PITSILLIDIS, A., LEVCHENKO, K., KREIBICH, C., KANICH, C., VOELKER, G., PAXSON, V., WEAVER, N., AND SAVAGE, S. Botnet Judo: Fighting Spam with Itself . In *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)* (San Diego, CA, USA, March 2010).

[19] QIAN, Z., MAO, Z. M., XIE, Y., AND YU, F. On Network-level Clusters for Spam Detection . In *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)* (San Diego, CA, USA, March 2010).

[20] QUINLAN, J. R. Ross quinlan's personal homepage. `http://www.rulequest.com/Personal/`.

[21] QUINLAN, J. R. Induction of decision trees. *Mach. Learn. 1* (March 1986), 81–106.

[22] RAMACHANDRAN, A., FEAMSTER, N., AND VEMPALA, S. Filtering spam with behavioral blacklisting. In *Proceedings of the 14th ACM conference on Computer and communications security* (New York, NY, USA, 2007), CCS '07, ACM, pp. 342–351.

[23] SHIN, Y., GUPTA, M., AND MYERS, S. Prevalence and mitigation of forum spamming. In *Proceedings of IEEE International Conference on Computer Communicationi (INFOCOM)* (Shanghai, China, 2011), IEEE Computer Society.

[24] SONG, J., LEE, S., AND KIM, J. Spam filtering in twitter using sender-receiver relationship. In *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID'11))* (September 2011).

[25] STEIN, T., CHEN, E., AND MANGLA, K. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems (SNS'11)* (New York, NY, USA, 2011), ACM.

[26] STRINGHINI, G., KRUEGEL, C., AND VIGNA, G. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference* (New York, NY, USA, 2010), ACSAC '10, ACM, pp. 1–9.

[27] THOMAS, K., GRIER, C., MA, J., PAXSON, V., AND SONG, D. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 2011).

[28] WEST, A. G., AVIV, A. J., CHANG, J., AND LEE, I. Spam mitigation using spatio-temporal reputations from blacklist history. In *Proceedings of the 26th Annual Computer Security Applications Conference* (New York, NY, USA, 2010), ACSAC '10, ACM, pp. 161–170.

[29] WILSON, C., BOE, B., SALA, A., PUTTASWAMY, K. P., AND ZHAO, B. Y. User interactions in social networks and their implications. In *Proceedings of the ACM European conference on Computer systems* (2009).

[30] XIE, Y., YU, F., ACHAN, K., PANIGRAHY, R., HULTEN, G., AND OSIPKOV, I. Spamming botnets: signatures and characteristics. In *Proc. of SIGCOMM* (2008).

[31] YANG, C., HARKREADER, R., AND GU, G. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID'11))* (September 2011).

[32] YARDI, S., ROMERO, D., SCHOENEBECK, G., AND BOYD, D. Detecting spam in a twitter network. *First Monday 15*, 1 (2010).

[33] ZADROZNY, B., LANGFORD, J., AND ABE, N. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the Third IEEE International Conference on Data Mining* (Washington, DC, USA, 2003), ICDM '03, IEEE Computer Society, pp. 435–.

[34] ZHAO, Y., XIE, Y., YU, F., KE, Q., YU, Y., CHEN, Y., AND GILLUM, E. Botgraph: large scale spamming botnet detection. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation* (Berkeley, CA, USA, 2009), USENIX Association, pp. 321–334.

[35] ZHUANG, L., DUNAGAN, J., SIMON, D. R., WANG, H. J., AND TYGAR, J. D. Characterizing botnets from email spam records. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats* (Berkeley, CA, USA, 2008), USENIX Association, pp. 2:1–2:9.

[36] ZOU, C. C., TOWSLEY, D., AND GONG, W. Modeling and simulation study of the propagation and defense of internet e-mail worms. *IEEE Trans. Dependable Secur. Comput. 4* (April 2007), 105–118.