CUD: Crowdsourcing for URL Spam Detection

PaperID 75

Abstract

The prevalence of spam URLs in Internet services, such as email, social networks, blogs and online forums has become a serious problem. These spam URLs host spam advertisements, phishing attempts, and malwares, which are harmful for normal users. Existing URL blacklist approaches offer limited protection. Although recent machine learning based URL classification approaches demonstrate good accuracy and reasonable throughput, they are based on observations from existing spam URLs and hard to detect new spam URLs when attackers employ new strategies.

In this paper, we present CUD (Crowdsourcing for URL spam detection) as a supplement of existing detection tools. CUD leverages human intelligence for URL classification through crowdsourcing. CUD crawls existing user comments about spam URLs already on the Internet, and employs sentiment analysis from nature language processing to analyze the user comments automatically for detecting spam URLs. Since CUD does not using features directly associated with the URLs and their landing pages, it is more robust when attackers change their strategies. Through evaluation, we find up to 70% of URLs have user comments online. CUD achieves an accuracy of 86.8% in terms of true positive rate with a false positive rate 0.9%. Moreover, about 75% of spam URLs CUD detects are missed by other approaches. Therefore, CUD can be used as a good complement to other approaches.

1 Introduction

The Internet today pervades every aspect of people’s daily lives. A wide variety of web services, like online social networks and video sharing services, attract millions of users every day [16, 20, 21]. Accompanying the massive proliferation of online services, cyber criminals quickly find their place. To host spam (including malware), attackers either create spam web services or comprise existing web sites, and then lure the victims to visit the Uniform Resource Locators (URLs) that uniquely point to the spam content. This approach has been extensively adopted to phish normal users, to propagate malware and to advertise solicitations such as pharmaceuticals to a large population of Internet users. Since the size of Internet is immense (more than 40 billion web pages [1]), it is very hard to detect the spam URLs at the time of their creation in realtime. Therefore, an effective defense relies on the identification of the attackers’ attempts to promote spam URLs. In other words, to identify whether a URL in a message (e.g., emails, Facebook wall posts, tweets) is spam in realtime.

One common approach is to use URL blacklists for filtering the spam URLs. Multiple URL blacklists accept queries from the public and return lookup results [6, 10–13]. Recently, researchers propose to use machine learning based approaches [24, 40, 46] to improve URL blacklisting. Some focus on the lexical properties of URLs [29]. Some resort to deep analysis on the URL’s landing page [37, 46] and hosting properties of domains [18, 31–34]. Usually, the features used in those approaches are based on observations made from existing spam corpses; thus, they might not capture the characteristics of new spam URLs when attackers change their strategies. Moreover, those approaches are useful to detect the spam URLs similar to existing corpses, and are less “intelligent” for detecting new spam URLs.

On the other hand, humans are good at recognizing even very sophisticated spam content. Most people encounter new spam URLs everyday even though many online services have already deliberately applied
URL blacklisting. With the advent of Web 2.0, more and more user-generated content becomes available online including users’ opinions and comments about the spam URLs. These user comments are potential valuable resources for identifying spam URLs. In this paper, we present a novel approach to detect spam URLs based on Crowdsourcing [15] and sentiment analysis [36] from nature language processing. Our approach complements the existing solutions.

The basic idea is to leverage the existing user comments about spam URLs already on the Internet to judge whether a URL is spam. Although each individual comment might not be highly reliable, aggregating them will result in an accurate decision on the classification of a URL. We take a passive crowdsourcing based approach by collecting the user comments about URLs already on the Internet through search engines. From the results returned by search engines, we identify the web pages that contain user comments about URLs. Then, we apply sentiment analysis on those user comments to see whether most users have negative opinions about the URLs. If the answers are yes, we classify the URLs as spam. Via nature language processing, we identify the nouns, verbs and adjectives in the comments and their semantic orientation (positive/negative). With this information, we employ four salient sentiment features. They all stem from consistent use of similarly formed statements in the user comments about the URL. These features grasp the commonalities people post their comment of the URL, e.g. they always choose nouns and adjectives (such as spam, malicious) and verbs (such as do not open, do not trust) to share their experience. Using supervised learning, we train a sentiment classification model based on label URLs for which we know they are spam or benign. Then, we use the model to detect whether a submitted URL is spam.

Comparing with existing machine-learning based spam URL filtering approaches, our design has the following advantages.

- One fundamental difficulty for most existing approaches is that some spam web pages can be easily identified by human beings but are very hard for computer programs. With a crowdsourcing approach, we leverage humans’ intelligence collectively to judge whether a URL is spam. By taking the advantage of human intelligence, our approach is able to detect spam URLs that otherwise are hard to be detected by existing automated approaches. Therefore, our approach can be a good complement for existing approaches.

- Existing machine-learning based URL-filtering approaches compute the classification result based on features from the URL such as lexical properties, the landing page, and the domain/IP information etc. If the attackers change their strategies on generating the URL and the landing page, sometimes, the detection systems need to add new features that can capture the characteristics of the new spam URLs. As a result, the detection systems need to keep their features up to date for maintaining good performance. Furthermore, even the features do not need to be changed, sometimes, for new spam URLs the parameters related to the features need to be adjusted for maintaining the accuracy on detect new spam URLs. In both cases, the detection systems have to be re-trained with new features or new spam training sets. On the other hand, although our approach use supervised learning as well, our features are based on user comments in a human language. Unless the language evolves, we do not need to re-train the model often. In other words, our approach has better stationarity.

- Unlike the URL itself or the landing page, user comments about the URL is not directly controlled by the attackers. Therefore, to some extent, our approach is more robust against evasion from the detection.

One problem needs to be aware is that not all URLs have user comments online. However, as we show in the experiments, this approach is applicable to a significant portion of URLs. In the three different URL datasets we collected from Facebook wall posts, the coverage is more than 70% for each dataset. In addition, based on our study, spam URLs have a much larger chance to have comments than the benign
ones. Therefore, we believe our approach can be a good complement for existing approaches. Moreover, the coverage can be further improved if we actively post the URLs with no user comments online and let users comment on them, which is a form of active crowdsourcing.

Furthermore, through the evaluation, using URL datasets collected from users’ Facebook wall posts with 27,651 URLs, we find our approach is accurate and fast, which can be used for realtime URL filtering. The true positive rate for the classification is 86.8% with a false positive rate 0.9%. In addition, on a 2.5GHz machine, we need about 7.682 seconds to classify a URL including crawling and processing the related user comments, the sentiment analysis, and classification. Furthermore, about 75% of spam URLs CUD detects are missed by other approaches. Therefore, CUD can be used as a good complement to other approaches.

In summary, we frame our contributions as:

I We design, implement and evaluate CUD, a realtime web URL classification system for spam URL detection based on reputation learned from users’ collective judgment on the URLs automatically.

II To the best of our knowledge, we are the first to apply crowdsourcing and sentiment analysis on user comments about URLs for spam URL detection.

III When designing the sentiment features, we consider the sentiment of verbs in addition to the adjectives used in sentiment analysis traditionally, because in the user comments about spam users tend to suggest others do not carry on certain actions such as click to the URLs.

The rest of the paper is organized as follows. We first provide our motivation and necessary terminology in Section 2. Section 3 illustrates the detailed system design, followed by a thorough set of experiments in Section 5. We discuss the related work in Section 6 and finally discuss in Section 7 and conclude the paper in Section 8.

2 Overview

In this section, we provide an overview of our design and define most concepts we are going to use throughout the paper.

The URLs are misused frequently in today’s network. They may be directed to spam, phishing and malware. Similar to previous work [40], in our study, we define the term “spam” to include: (i) advertisements such as for pharmaceuticals or for adult content, (ii) phishing attempts, and (iii) web pages containing malware.

We would like to design a spam URL detection system that take a URL as input and detect whether the URL is spam in realtime. Instead of using the characteristics directly associated with the URL, we leverage users’ comments about the URL to make the decision. We believe human beings are very good at detect sophisticated spam URLs especially if we take the advantage of the collective wisdom from multiple people.

Crowdsourcing is a distributed problem-solving and production model [15]. Here we mainly employ one type of the crowdsourcing crowdwisdom [15] (also called collective intelligence [30]). Crowdwisdom means “a form of universally distributed intelligence, constantly enhanced, coordinated in real time, and resulting in the effective mobilization of skills”. “Since no one knows everything, everyone knows something, all knowledge resides in humanity.” [30] In classical crowdsourcing (active crowdsourcing), the tasks are predefined, and people contribute to the tasks voluntarily. In this form, people actively participate in the crowdsourcing effort. On the other hand, in this paper, we mainly employ passive crowdsourcing, in which case we crawl the user comments about a URL, which have already existed on the Internet. Nevertheless, we find active crowdsourcing, i.e., posting URLs without user comments on the online forums to encourage people to comment on them, can further help improve the coverage of our approach.
There is a huge amount of user-generated content on the Internet including user opinions about URLs. One challenge is how to identify the user comments about URLs from the enormous information available on the web. To this end, we first leverage search engines to collect the web pages related to the given URL or its domain. Then, we identify the pages related to user-generated content such as from blogs or online forums. Finally, we leverage nature language processing techniques to identify the judgment about the URL is positive or negative based on four sentiment features.

To some extent, our design is to mimic what people do when they want to know the opinions from other people online about a subject. For instance, we may search the reviews about a product using search engines. Actually, sometimes, when we receive an unknown URL, we might search the URL in search engines to check what others say. Our approach is to make this process fully automatic.

One concern about this approach could be whether user comments online are trustworthy. We find both the results from other researchers and our own results confirmed that this approach is indeed reliable. In [39] the authors collected a large amount of data from YahooAnswer to study how effective and satisfactory information they provide to the information seekers. They demonstrated that “interrogators were not only satisfied with a given answer; they also found these selected answers to be of high quality.” Similarly, in our study (results in Section 5), we also obtained the accuracy of about 87% with less than 0.9% false positives. Furthermore, since our sentiment features are calculated from all the posts related to the URL, they are more reliable than those from single user’s comments.

Wednesday, March 5, 2008

**Spam/Solicitation in the future**

So, I remember a while back I had to text a number on my cell phone to ensure that I was put on a "no solicitation" list. I remember being completely in awe that corporations would even be allowed to do such a thing - cell phones are hyper-personal, more so than land line phones. I mean, my cell phone number essentially acts as my second Social Security number - a number assigned specifically to me. What right does anyone have to retrieve that information without my knowledge?

Well, lo and behold, a couple days ago I got a spam message - not on my cellphone, not via e-mail, but - through Facebook. I don't know if this new or not, but it's certainly the first I've heard of it. I got a message on my wall from one of my "friends" (someone who I have not talked to in years, which, in itself says quite a bit about Facebook) saying "Sure! I get my ringtones from www. RingRockstar.com". The correct capitalizations and punctuation are a dead give away that this person did not in fact write this message.

What's most detestable is the phantom approach - I got an e-mail saying THIS girl WROTE this on MY wall - not RingRockstar.com writing on my wall, but a "friend". How long before Facebook and other websites fall victim to these schemes?

Eric

Posted by the COOL class at 5:30 AM

---

In this study, we found most user-generated content mainly came from blogs and forums. The human generated blog and forum posts are the information source that we use to decide whether an input URL is spam. Blog and forum posts both contain a title and a body of post content (which is called “post” from now on for simplicity). Figure 1 and Figure 2 provide two examples for them, respectively. The format of blog and forum posts differs slightly. A blog post is often generated by one person. The person makes a statement in the title and explains his statement in the post. In contrast, a forum post is essentially a thread of
If you go to www.7g214.info, is it a virus?

i got a facebook post in the form of an advertisement thing from one of my friends, it included this site to go to

it looks kind of weird but im not sure

Is it legit or not?

3 years ago

Best Answer - Chosen by Voters

SiteAdvisor has no listing for the site and it doesn't look like one I personally would trust.

One way I check is to run the site through McAfee SiteAdvisor. It is a free extension for both Internet Explorer and Firefox that checks websites for bad stuff.

Another thing to look at is the name of the site. Does www.7g214.info sound legit to you? There are a few indicators.

1) The 7g214 has little meaning that can be applied to it. Most sites have a theme that even their name corresponds to.

2) .info sites are often cheap and easy to obtain. This is usually a concept applied by malware sites to save a few bucks.

3 years ago

Figure 2: A forum post example.
discussion. Its title is usually a question about the URL. Its post is generated by multiple other people trying to answer the question. Despite the difference, both blog and forum posts contain the statements about the nature of the URL (whether it is spam) and are processed identically by our system.

Given an input URL, we rely on querying search engines to identify the related web pages that will be analyzed by our system. In the result page returned by a search engine, each entry is called a search engine hit. We further decompose a search engine hit into three basic components: the title, the snippet and the information locator. An illustrative example is presented in Figure 3. The information locator is essentially the URL of the web page that contains information about the input URL. We avoid directly calling it a “URL” again so as not to be confused with the input URL. The given example also contains a small part showing the number of posts. Not all search engine hits contain such information, but it has significant meaning when we select the information locator pointing to blog or forum pages later.

3 System Design

In this section, we present the detailed system design, starting with the system architecture in Section 3.1. After that, we present the four major components: the crowdsourcing crawler in Section 3.2, the information collector in Section 3.3, the posts analysis engine in Section 3.4 and the classifier in Section 3.5.

3.1 System Architecture

We face two grand challenges in the design of our system. First, billions of web pages exist in the wild. The system needs to accurately locate those that contain the users’ comments on certain URLs. Second, the comments are not organized in any structured format, but are in freeform human language. The system needs to comprehend the semantics correctly. To address the first challenge, we exploit the functionality of search engines. The search engines visit, analyze and index billions of web pages on the Internet. Although none of them claims to have indexed all the web pages, querying a combination of them provides a nearly complete set of web pages that are relevant to the queried terms. To address the second challenge, we propose a novel set of features and adopt a collection of state-of-the-art semantic analysis tools. The features we propose are based on our observation that people usually use nouns and adjectives, such as “spam” and “malicious”, as
Figure 4: The system design

well as verbs, such as “do not click” and “do not trust” to comment on a certain URL.

Figure 4 shows the workflow for classifying a URL. Our system accepts URLs as input and queries search engines to locate the human submitted comments about the URLs on the Internet. After that, it collects and combines the comments. It then transforms the raw information into meaningful Boolean and real-valued features and provides these results to a trained classifier, outputting the classification result finally. The result indicates whether the input URL is spam or not. Accordingly, the system has four major components: the crowdsourcing crawler, the information collector, the posts analysis engine and the classifier. The crowdsourcing crawler uses the input URL to query well-known search engines, such as Google, Yahoo, and Bing, and fetches the search hits. Based on the results returned by the search engines, it extracts the information locators of all the web pages that are relevant to the input URL. After that, the information collector crawls the actual web pages using the information locators. Note that not all web pages are useful. Rather, the system only analyzes the forum and blog posts. Consequently, the information collector filters out other web pages, such as the domain registration information of the input URLs. The posts analysis engine is given a set of human-generated posts that comment on the input URL. It leverages semantic analysis tools and supervised machine learning to identify the semantic orientation of these posts and aggregate them into a feature vector representing the input URL. Finally, the classifier produces the final classification result for the input URL.

3.2 Crowdsourcing Crawler

The key idea behind the system is to combine the human generated comments on URLs strategically. The goal of the crowdsourcing crawler is to identify the information locators pointing to useful web pages that contain such comments. Unfortunately, the number of web pages existing today is enormous, which makes it difficult to locate the useful ones.

The crowdsourcing crawler exploits the functionality of search engines to achieve this goal. If a search engine receives a URL, which we denote as the “input URL”, as the query term, it will return the snippets of all web pages indexed by the input URL, along with the URLs of these indexed web pages, which we denote as the “information locators” as stated before. The crowdsourcing crawler submits the URL to query the search engine and extracts the returned information locators. Assuming the search engine is doing a good job of crawling the Internet, the extracted information locators should have good coverage, in the sense that most web pages with comments on the URLs are obtained.

3.3 Information Collector

In the previous stage, the crowdsourcing crawler has obtained a set of information locators. The information collector accesses the actual web pages via the information locators and fetches the comments. Note that
it is not a simple crawler. Recall that the crowdsourcing crawler extracts information locators returned by
the search engines without any discretion, whereas the detection of URL Spam is based only on people’s
comments. As a result, the information collector accomplishes three tasks: i) to identify the information
locators pointing to forum and blog web pages, ii) to extract the title and content of each piece of comment
on the web page, and iii) to filter the extracted comments to wipe out noises.

To accomplish the first task, we develop a filter that implements a series of heuristics to match the
characteristics of forum and blog web pages. The filter discards all the irrelevant web pages. We defer the
description of the heuristics that we adapt to in Section 4.

After the first step of filtering, the information collector crawls the remaining web pages and extracts
the title and content of the comments in the web pages. This is challenging because the crawled pages are
essentially text files written in the HTML language with very complex layout. There is no rule that regulates
which specific HTML tags mark the starting and ending position of the actual comment in the HTML file.
We overcome this difficulty by identifying the common structure shared by most forums. After the structure
of the web pages is identified, the actual comments can be located and extracted accordingly.

Finally, the information collector performs another pass of noise filtering to the collected comments.
Two types of noises are the target of the final filtering, both of which are comments in forums and blogs
with the actual content totally unrelated to the input URL. The first type is the peripheral content, such as
hypertext links and advertisements. The second type is the forum spam [5], which is the spam posted onto
the forums. We developed a set of rules for the noise filter. We defer the detailed description of our rules in
Section 4.

3.4 Posts Analysis Engine

Our system detects URL Spam by determining the emotion of the linguistic text of the crawled comments,
which can be casted as a multi-class classification problem. In our case, we consider three emotion classes.
Let $E$ denote the emotion classes, such that $E = E_0, E-, E+$. $E_0$ denotes the special case of neutrality, or
absence of emotion. $E-$ denotes that the text describes the URL as bad. $E+$ denotes that the text describe
the URL as good. Let $T$ denote the text. The goal is to determine a mapping function $f : T \rightarrow E$

The classification of emotion is based on the semantic orientation of words that comprise the text, which
is known as sentiment analysis [25, 27, 28]. However, our task of spam URL classification is distinct from
the existing approaches. The key difference is that most of the existing approaches only use adjectives to
infer the emotion of the text. In contrast, our system analyzes nouns and verbs in addition to adjectives. The
reason is that people may express their emotion to the target URL using verbs, such as “do not enter” and
“do not trust”, which is special for the comments on URLs.

3.4.1 Sentiment Analysis

The basic idea is to compute word’s semantic orientation based on a list of seed negative words, which
are known to carry negative emotion and are thus used as the ground truth. We match the nouns and
adjectives in $T$ against the negative word list to determine their semantic orientation. Note that we are not
performing straightforward exact string matching at this stage, which would result in high inaccuracy due
to the complexity of human language. In particular, three factors need to be handled to ensure the accurate
computation of word orientation, which are i) synonyms, ii) words with the same root, and iii) negative
expression of the words (for example: not legal).

We need to handle synonyms because they share the same semantic orientation. If a word, $w$, in $T$ is the
synonym of some word in the negative word list, apparently $w$ should also carry negative orientation. Exact
string matching, however, cannot produce this result. To solve this problem, we expand $w$, as well as the
negative word list, into the set of their synonyms, respectively. If these two sets intersect with each other, we can determine that \( w \) carries negative orientation.

Words with the same root cause problems due to the same reason as the synonyms. For example, if the word "spam" is in the negative word list, the words with the same root, such as "spammer" and "spamming", should also carry negative orientation. The solution we adopt is to compare the root of each word in \( T \) with the roots of the negative word list after the expansion with synonyms.

Negative expression of the words, as the name indicates, has the ability to reverse the orientation of the word after it. For example, “not illegal” expresses positive orientation while “illegal” is negative. As a result, if any word is determined to have negative orientation, we check if a negating word precedes it. If so, we negate its orientation into positive.

3.4.2 Feature Extraction

One of the most important tasks for supervised machine learning is to devise a set of effective features that can distinguish different classes. We propose two novel features, strong negative word ratio and negative verb, based on our observation of the particular task of URL classification. We also adopt two commonly used features in text orientation classification, which are negative word count and positive word count. We use the combination of the four features to train the classifier.

**Strong negative word ratio.** Recall that we have multiple posts commenting the target URL. The feature of strong negative word ratio is calculated as the proportion of posts that hit at least once in the strong negative word list, which include “spam”, “malicious”, “illegal”, “malware”, etc.

**Negative verb.** We have stressed that the usage of verb to express orientation is a unique feature pertaining to the comments on URLs. We observe that this feature is particularly determinant. For example, with very high probability a document will carry negative emotion if it contains a negative verb such as “do not open”, regardless of other features. As a result, if any post hits our verb list, which includes “open”, “click”, “enter” and “check out”, etc., with a preceding negating word, this feature will be evaluated as “True” for the input URL.

**Negative word count and positive word count.** These two features are calculated as the sum of negative and positive word hit by all the posts commenting on the input URL, respectively.

3.5 Classifier

In the research community, the dominant approach for sentiment analysis is based on machine learning techniques. Comparing with the knowledge engineering approaches that require manual definition of a classifier by domain experts, the machine learning techniques have the advantage of significantly saving the expert labor power. Hence, in our system we use supervised machine learning to train the classifier. In particular, we choose to use Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [17] as the training algorithm. It was proposed as an optimized version of IREP [22]. RIPPER builds a ruleset by repeatedly adding rules to an empty ruleset until all positive examples are covered. After the ruleset is constructed, an optimization pass massages the ruleset to reduce its size and improve its fit to the training data. We also tested other training algorithms but RIPPER provides the best accuracy.

4 Implementation

Our system implementation exploits multiple existing tools and libraries. We now introduce the details of various system components.
4.1 Crowdsourcing Crawler

We believe that none of the well-known search engines can crawl the Internet completely. As a result, the crowdsourcing crawler queries multiple search engines to crawl the web pages related to the input URL to our best effort. The search engines that are queried include Google, Yahoo and Bing. We use Xgoogle [14], a Python wrapper library, to interact with Google search engine. We use Bingapi.py [4], a thin python wrapper over the Bing API, to query Bing search engine. For Yahoo, we use Yahoo python API [9] to harvest the results.

Figure 5 illustrates the detailed process. For an input URL, we first pass it into Xgoogle, Bing API and Yahoo API to get the top 100 search hits of each search engine. If there is no information returned by querying the URL, we use the domain name as the query term instead. Then we compare and combine the results, deleting the duplicated ones to prepare the information locators for further processing.

We plot the number of forum and blog posts return from each search engine and show the result in Figure 6. In this study, we randomly select 7468 URLs from the dataset we used, the detailed description of which is available in Section 5. We query all the URLs to each search engine. Google has the best coverage. Nearly 60% of the URLs have the blog and forum information returned. However, no single search engine can provide full coverage of these URLs. Using the combination of all the three search engines results in the best coverage, which is about 70%, as we will show in Section 5.

4.2 Information Collector

To implement the information collector, we build a multi-threaded web crawler. We also use Beautiful-soup [3] to get structured layout of the forum and blog web pages to facilitate our extraction of the actual post information.

In addition, the information collector contains two filters. One filters out the search engine hits that are not blog or forum pages, while the other one filters out noises in the comments obtained from the blog and forum pages. We now describe the rules that these filters adopt in detail.

**Search hit filter.** We examine the search hits obtained from the crowdsourcing crawler to identify the useful information locators using a series of heuristics. If a search hit satisfies any one of them, it will be recognized as a forum or blog hit. The corresponding information locator will be stored and accessed later. The heuristics are:

- The title of the hit contains the input URL and is a question.
The number of forum and blog hits

Figure 6: The CDF of number of forum and blog posts returned from each search engine.

- The title of the hit contains at least one keyword in the list that is associated with the “forum” website class in [41].
- The hit shows the number of posts, which is a special effect only pertaining to the websites that the search engine recognized as forums.
- The snippet in the search hit contains keywords including “blog”, “forum”, “group”, etc.
- The information locator in the search hit is in the top 100 famous blog and forum list [2].

Noise filter. The noise filter first delete all the “<a href>” HTML tags to get rid of the hypertext. After that, it adopts the following rules to select the comments relevant to the input URLs:

- The title contains the input URL.
- The title contains a strong negative keyword, such as “malicious”, “spam”, etc.
- The title contains “Facebook”, “myspace”, “MSN”, “IM”, “Message”, etc.

The content will be kept if any of the above rules is satisfied. Otherwise, the filter will deem it as noise and discard it. These rules help us get a set of clean comments for further analysis.
4.3 Posts Analysis Engine

In the posts analysis engine, we use the Tokenizer class in NLTK [8] to split original text $T$ and collect all nouns, verbs and adjectives in it. Next, we generate the synonym set for each noun and adjective with the help of WordNet [35], and compute their semantic orientation. For the verbs, we devise a list of verbs that expresses strong negative emotion and record whether the post hits the negative verb list. After this preprocessing, we use supervised machine learning to classify the emotion of the text $T$.

4.4 Classifier

We use the Weka [26] tool as the framework for our supervised machine learning method. We choose Jrip as our classifier, which is the implementation of the RIPPER algorithm.

5 Evaluation

In this section, we systematically evaluate the performance of our system. We first introduce the two datasets used for the evaluation in Section 5.1. Our system requires the blog or forum posts that comment on the input URL. We show that more than 70% of the URLs tested satisfy this requirement in Section 5.2. After that, we use standard machine learning tools to test the detection accuracy in Section 5.3. We measure the runtime performance in Section 5.4. We also compare with multiple well-known blacklists and show that about 74% of the malicious URLs detected by our system are missed by the blacklists in Section 5.5. Finally, we present our findings like where in the Internet the most comments are found in Section 5.6.

5.1 Dataset

We use two datasets to evaluate our system. The first one (Facebook1) comes from the result of a previous study [23]. In this study, 187 million wall posts generated during the period from January 2008 to September 2009 are crawled. 1.1 million unique URLs are extracted. 14461 URLs are labeled as “malicious” using a collection of approaches. The rest URLs are labeled as “benign”. We use all the malicious URLs and 8030 randomly selected benign URLs in the first dataset.

The second dataset (Facebook2) is also crawled from Facebook, but is more recent than the first one. In April 2011 we crawl five public Facebook group profiles, collect all the wall posts that are visible and extract all the embedded URLs. This dataset contains 5160 URLs in total, all of which are posted during the period from January 2011 to April 2011. However, this dataset contains a mixture of malicious and benign URLs and is not labeled. Table 1 gives out a summary of both datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time</th>
<th>URL #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook1</td>
<td>01/2008 - 09/2009</td>
<td>22491 labeled</td>
</tr>
<tr>
<td>Facebook2</td>
<td>01/2011 - 04/2011</td>
<td>5160 unlabeled</td>
</tr>
</tbody>
</table>

Table 1: The summary of the two datasets.

5.2 Coverage

Our system relies on the forum and blog posts about an input URL to classify it. An important concern is how often an input URL would have such information available. We define the detection coverage as the ratio of the number of the input URLs that have the blog and forum information using our crowdsourcing crawler to total number of the input URLs. Table 2 gives out the result. For all types of URLs, the coverage
Table 2: The coverage of two datasets used in this study.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coverage</th>
<th>Have valid feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook1 (malicious)</td>
<td>98.2%</td>
<td>81.4%</td>
</tr>
<tr>
<td>Facebook1 (benign)</td>
<td>79.0%</td>
<td>63.5%</td>
</tr>
<tr>
<td>Facebook2</td>
<td>71.4%</td>
<td>34.6%</td>
</tr>
</tbody>
</table>

Table 3: Results for training on data with different benign to malicious ratios.

<table>
<thead>
<tr>
<th>Benign URL #</th>
<th>Malicious URL #</th>
<th>TPR</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2500</td>
<td>2500</td>
<td>87.4%</td>
<td>0.6%</td>
</tr>
<tr>
<td>2500</td>
<td>5000</td>
<td>86.6%</td>
<td>0.3%</td>
</tr>
<tr>
<td>5000</td>
<td>2500</td>
<td>86.6%</td>
<td>0.4%</td>
</tr>
</tbody>
</table>

is above 70%. This serves as a strong basis for our follow-up analysis. We also observe that the coverage is higher in the Facebook1 dataset, which is older, and that the coverage is higher for the malicious URLs than the benign ones. This is likely because people prefer to discuss the malicious URLs on the Internet more often. When people encounter a strange URL, they may find it suspicious and post this URL onto public forums to seek advice from others.

The system essentially classifies the semantic orientation of the forum and blog posts as either positive or negative. However, all the texts do not necessarily contain such an orientation. Instead, some of them may be neutral comments, and will not help the classifier make the decision. The posts analysis engine will produce feature vectors comprising only “0” and “false” for the texts without any orientation. Accordingly, they are excluded in both the training and the testing process. Table 2 shows the number of URLs after the exclusion in the third column.

5.3 Accuracy

For any detection system, the high accuracy is required. 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 URLs correctly classified as malicious divided by the total number of malicious URLs. False positive rate reflects the detection error on the legitimate URLs. It is defined as the number of URLs incorrectly classified as malicious divided by the total number of legitimate URLs. In this section, we evaluate both the overall accuracy and the accuracy using different feature sets.

5.3.1 Choose Training Set

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 [45] and used by Thomas [40], we adjust the ratio of benign URLs to malicious URLs in the training set by randomly sampling to tailor the performance. We use the ratios of 1:1, 1:2 and 2:1. After adjusting the ratio, we run the JRip classifier using 10-fold cross validation. In 10-fold cross validation, the dataset is randomly divided into 10 parts of equal size. Then nine parts will be used as the training set while the remaining part will be used as the testing set. This process is repeated for 10 times so that each part will be used as the testing set once. Table 3 shows the detection accuracy.

The true positive rate varies slightly from 86.6% to 87.4%, while the false positive rate varies from 0.3% to 0.6%. We choose 2500:2500 URL set as our training set, because it yields a reasonable true positive rate
and false positive rate.

5.3.2 Overall Accuracy

We first test the overall detection accuracy. All four proposed features are used. As previously stated, we use 2500 benign URLs and 2500 malicious URLs from the dataset Facebook1 as the training set.

We conduct experiments on both datasets to test the overall detection accuracy. First, we use all the URLs in the dataset Facebook1 that are not included in the training set as the testing set. The true positive rate is 86.8% while the false positive rate is 0.9%. Next, we use the dataset Facebook2 as our second testing set. Among 5160 URLs contained in the dataset, 27 of them are classified as malicious by the system. Since this dataset is not labeled, the true positive rate and false positive rate cannot be evaluated directly. Instead, we manually validate these 27 URLs and confirm that 19 of them are correctly classified. In addition, we randomly select 250 URLs (about 5% of the facebook2 dataset) that are classified as benign and manually investigate them. Only 9 of them are actually malicious. This shows that our system can still get a reasonable accuracy on new dataset without retraining.

5.3.3 Accuracy of Different Feature Sets

As stated in Section 3, we divide our features into two sets: the URL related features and the general features. In order 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 4. We use the URL set described in Section 5.3.1 as the training set and the rest URLs in the Facebook1 dataset as the testing set. Note that our purpose is not to compare between these two feature sets. We observe that using either feature set would cause a relatively high false positive rate. However, using their combination would reduce the false positive rate to only 0.9% and improve the true positive rate to 86.8%.

5.4 Runtime Performance

In addition to the accuracy, the runtime performance is critical to our system. As shown in Figure 4, the whole process of classifying an input URL is done by four major components: the crowdsourcing crawler, the information collector, the document analysis engine and the classifier. Accordingly, we measure the time consumption of each component. We conduct the experiment on a Linux machine with 2.5GHz Xeon CPU and 16GB memory. We randomly select 150 input URLs and present the average result in Table 5. On average, it takes 7.682 seconds for the system to complete the whole process to decide whether an input URL is spam. The crowdsourcing crawler and the information collector need to fetch web pages from the Internet and are expected to be time-consuming. They take up about 37% of the total running time. The complex semantic analysis on the crawled documents also takes up about 63% of the total running time. Comparing with them, the running time consumed by the classifier is negligible.
5.5 Comparison with Other Tools

In the experiments, our system successfully detects 8054 spam URLs. Since CUD is designed to be used in combination with other tools, we test how many URLs that CUD can detect are missed by other tools, in order to measure the performance gain that CUD brings into the combination. The tools we tested are Google Safebrowsing [6], McAfee Siteadvisor [7], Surbl [12], Uribl [13], Spamhaus [10] and Squidguard [11]. All of them are URL blacklisting services. Google Safebrowsing provides APIs to query the underlying blacklist. McAfee Siteadvisor returns one of the four possible results for a given URL, which are “malicious”, “suspicious”, “benign” and “unknown”. If the result is either “malicious” or “suspicious”, we count the input URL as being detected successfully. Surbl, Uribl and Spamhaus accept queries via DNS request. Squidguard allows us to download the blacklisted URLs to match the input URLs. Table 6 shows the detection result of the listed tools. We aggregate the result of the last four blacklists as “other blacklists”. Among the URL blacklisting tools, McAfee Siteadvisor has the highest detection rate. Nonetheless, it only detects about 25% of the spam URLs. Google Safebrowsing and other blacklisting tools can only detect very few spam URLs. This result confirms that our system can be used to complement existing tools, or to be incorporated into existing tools to further boost the performance.

5.6 Findings

5.6.1 What Kind of URLs Are Detected

We hypothesized that URLs that are used more frequently by the attackers are more likely to be discussed by Internet users and would thus have a larger change to be detected by our system. We measure the frequency, meaning the number of appearance in the crawled wall posts, of the 8054 spam URLs that have been detected and compare that with all the URLs in the dataset. Figure 7 shows the CDFs of the frequency of the two types of URLs, respectively. We find that our hypothesis is not entirely true. Both types of URLs have about 50% of them appearing only once. However, for the URLs that have appeared for multiple times, the spam ones exhibit somewhat hither frequency.

5.6.2 Where Are the Answers from

Our approach processes people’s comments on the input URLs posted in forums and blogs. In forums, people often use the form of Q&A to discuss the URLs. In blogs people prefer to share the malicious URLs that they found and to give the explanation. In addition, one blog post often reveals multiple malicious URLs, especially when the owner is a security specialist or when the blog is hosted by a company in security business like Symantec.

In our experiments, the number of forum posts discussing the URLs has a highly skewed distribution across different sites. The 5 forums where we obtain the most comments on the URLs are “http://answers.yahoo.com”

<table>
<thead>
<tr>
<th>Blacklist Name</th>
<th>Google Safebrowsing</th>
<th>McAfee Siteadvisor</th>
<th>Other Blacklists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detected URL #</td>
<td>7</td>
<td>2069</td>
<td>31</td>
</tr>
</tbody>
</table>

Table 6: The existing tools’ detection result on the 8054 spam URLs that are detected by our system.
The frequency of the URLs in the wall post 

CDF

All URLs
URLs have been detected

The frequency of the detected URLs.

"http://answers.com", "http://wiki.answers.com", "http://askville.amazon.com/index.do" and "http://aolanswers.com". They altogether contain about 47.1% of the total number of forum posts that our system has collected. For blog posts the two heavy hitters are "http://belsec.skynetblogs.be" and "www.blogger.com".

6 Related Work

We discuss prior related work by organizing them into three general categories: studies of spam URL detection, crowdsourcing and sentiment analysis.

Studies of spam URL detection. Most prior studies of spam URL detection rely on deep analysis of the URL’s landing page. A popular approach is to first build the landing pages into feature vectors, using a feature set including HTML forms, embedded links, word count and others, then to resort to supervised machine learning to classify the vectors [18, 24, 31, 37, 40]. In comparison, our system avoids the content downloading of landing pages, which is strictly safer. McGrath et al. studies the lexical features, registration and host information of Phishing URLs [34]. Ma et al. propose to use the URL lexical features and the host-based features to train the classifier [32, 33]. Execution-based approaches have been proposed to detect malicious URLs conducting drive-by-download attacks [19, 38, 43]. These approaches render the landing page, execute all the contained scripts in virtual machines and catch the abnormal activity during
the execution. In addition, Xie et al. propose to only use the URL lexical information to extract regular expression signatures for spam URL detection [44]. This approach, however, is limited by the expressive power of regular expression and may fail to produce meaningful signatures if the attacker introduces excessive randomness into the URLs. All the existing work faces the continuing arms race with the attacker and need to adapt to the newest features quickly. In comparison, our system is based on blog and forum posts generated by the Internet users that the attacker has little control on. The way that the general Internet user expresses the “positive” and “negative” emotion is relatively stable. Hence, our system does not need frequent re-training to maintain the high detection accuracy.

Studies using crowdsourcing. Crowdsourcing is an online, distributed problem-solving and production model that has emerged in recent years [15]. The basic underlying assumption is that no one knows everything, everyone knows something, and that all knowledge resides in humanity. Given all the knowledge that exists in the wild, how to aggregate it for practical usage is a great challenge. Trestian et al. propose to query the Google search engine to classify Internet endpoints [41], which is the most closely related work to ours. Although both works rely on crowdsourcing, the goal and the way of processing the gathered information are very different. While the previous work mostly use keyword based matching for the classification, we perform complex semantic analysis to extract and aggregate the sentiment of the Internet users.

Studies of sentiment analysis. Sentiment analysis has been extensively studied recently because of the commercial interest. Hu et al. [27] query online documents for positive and negative opinions of various product features, such as the size of a digital camera. Glance [25] and Hurst [28] both make use of manual determination of polarized adjectives for a specific domain. The research can also be divided into approaches focusing on the sentence or clause level [25, 27], and the ones focusing on the document level [42]. Our work distinguishes from existing works in multiple ways. First, we consider the sentiment of verbs in addition to consider the adjectives only. Second, we divide the negative words into strong negative words and normal negative words, which play different role in the sentiment analysis. Finally, sentiment analysis is only one step in the whole process of URL classification. It is based on the documents collected via crowdsourcing and its result is organized into feature vectors to feed the trained classifier.

7 Discussion

In this section, we discuss the limitations of CUD and potential attacks to the system. We also discuss how CUD can be used in combination with existing tools.

Limitations The most obvious limitation is that the input URLs must have people’s comments on the Internet. Not all the URLs have this information. In our implementation, the system uses the comments on the domain name if no comments on the full URL can be found. It will cause false negatives if the attacker misuses legal domain names to spread the malicious content. The two common ways to do this are compromising legitimate websites and using URL shortening services. We expect the first case to happen very infrequently. We defer the discussion of the URL shortening service to the attack resilience analysis. Another way to overcome this limitation is to actively post questions about the URLs onto the forums in order to trigger people to comment on them. In our preliminary experiment, the active posting turns out to be effective. We receive people’s comment on the posted URLs within one day. How to carry out active posting in large scale is part of our future work.

The second limitation is posed by the search engine service providers. For example, Google blocks an IP address if it generates search queries too frequently. As a result, we halt the system for a few seconds after each query to Google. This limitation essentially puts a cap on the number of URLs that our system can classify in a given time period. There are two ways to address this limitation. The first one is to use the paid search service instead of the free one to get a much higher quota. The second one is to cache the search result of the input URLs locally.
**Attack Resilience**  We consider two types of attacks to the system. The first type is to tamper with the people’s comments on the URLs. The attacker can either make positive comments on the malicious URLs or make negative comments on the legitimate URLs. However, the attacker does not have control on the comments made by the normal Internet users. Since CUD aggregates all the comments on the input URLs, the attacker essentially needs to outnumber the rest of the Internet users in the number of comments in order to flip CUD’s detection result. It is very hard to be done, since well-known forums usually enforce very strict policies to restrict the number of posts that a user can make. In addition, the cost for making large number of posts is high.

The second type of attack is to hide the real malicious URL behind URL shortening services via redirection. Currently CUD does not trace through the redirection links, but it is not technically difficult to implement such functionality so that CUD can always work on the landing URLs.

**Deployment**  Because CUD cannot classify URLs without people’s comment, a stand-alone deployment of CUD is not preferred. Rather, CUD can be deployed in combination with other tools to complement them. The benefit is twofold: *i*) It can detect significant number of malicious URLs that will be otherwise missed by other tools. *ii*) The approaches focusing on the landing pages constantly need fresh training set to maintain the high detection rate. The newly emerged URLs with carefully crafted features are likely to evade such detection systems. However, CUD can catch them since the way the general people comment on the URLs stays relatively stable and the attacker has very little influence on it.

### 8 Conclusion

In this paper, we propose a novel approach that uses crowdsourcing to detect spam URLs. It distinguishes from existing approaches that it collects and processes human knowledge about URLs on the Internet to compute the nature of URLs, *i.e.*, whether or not they are spam URLs. We implement a prototype system called CUD, which passively collects human comments on URLs that already exist and leverage sentiment analysis to understand the semantic orientation of the human comments about the URLs. CUD can be used as a complementary tool to existing approaches, as about 75% of spam URLs CUD detected are missed by other approaches. It achieves 86.8% true positive rate with 0.9% false positive rate.

### References


