Spam Filtering Beyond Tokens

Oleg Kolesnikov, Wenke Lee, Richard Lipton, Steve R. Webb, and Calton Pu
{ok,wenke,rjl,webb,calton}@cc.gatech.edu

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Text-based spam filters (e.g., keyword and statistical learning filters) use tokens found in message content analysis to separate spam from legitimate messages. The effectiveness of these token-based filters is due to the presence of token *signatures* (i.e., tokens that are invariant for the many variants of each spam message). Unfortunately, it is relatively easy for spammers to hide or erase these signatures through simple techniques such as misspellings (to confuse keyword filters) and camouflage (i.e., mixed spam and legitimate content used to confuse statistical filters). Our hypothesis is that spam contains additional signatures that are more difficult to hide, and as a result, we believe tools should be developed to focus on those signatures. A concrete example of this type of signature is the presence of URLs in spam messages which induce contact by the victim. We use the syntactic constraints of URLs to find them in emails, and we use semantic knowledge and tools (e.g., search engines) to refine and sharpen the spam identification process. Preliminary experiments show high accuracy for this type of URL-based filtering.

1 Introduction

Spam filtering (i.e., distinguishing between spam and legitimate email messages) is a commonly accepted technique for dealing with spam. Current spam filters classify messages based mainly on the tokens found in the message text. This approach has had mixed results. On the one hand, many spam messages have token signatures that facilitate filtering. These signatures typically consist of tokens that are invariant for the many variants automatically generated by spammers. On the other hand, spammers use various techniques to defeat filters. For example, keyword filters can be defeated using deliberate misspellings [21], and statistical learning filters can be confused using camouflage (i.e., legitimate content added to spam messages) [20].

To overcome the limitations of token-based filters, we propose an approach that looks at other forms of signatures to complement existing text-based techniques. In this paper, we focus on spam messages which contain URLs. By checking the URLs and verifying a user's interest in the websites referenced by those URLs, we are able to add a new dimension to spam filtering. The key observation is that most spam messages contain URLs which are "live" since the spammers would not be able to profit without a functioning link to their site.

This position paper has two main contributions. First, we describe three techniques for filtering email messages that contain URLs: URL category whitelists, URL regular expression whitelists, and dynamic classification of websites. Second, we describe a prototype implementation of these three techniques and use a preliminary evaluation to show how spam messages can be successfully classified. Our results suggest that new dimensions in spam filtering (e.g., using URLs) deserve further exploration.

The remainder of the paper is structured as follows. Section 2 gives an overview of the related work done in this research area. In Section 3, we describe our approach. Section 4 discusses the details of our system's implementation, and Section 5 evaluates our approach against a popular text-based spam filter. We finally conclude in Section 6.

2 Related Work

Filtering is currently the standard approach used to stop email spam. However, most of the current approaches and products use content-based filtering. The content-based approaches include whitelisting, blacklisting, keyword-based [3], statistical classification [16, 1], heuristic-based filtering [18, 4], and collaborative filtering [15]. Other classes of filtering approaches include challenge-response [11], MTA/Gateway filtering (Tarproxy [14], greylisting [10], etc.), and micropayments [13, 19].

Some content-based approaches rely exclusively on message headers: automatic and Bayesian whitelisting [12], blacklisting (MAPS, RBL), and others. These techniques have two main disadvantages. Spammers can easily forge message headers, and legitimate domains can easily become blacklisted.

Other content-based approaches rely on message tokens and their corresponding statistics. For example, simple Bayesian Machine Learning approaches, introduced by Duda et al. [5] and applied to spam filtering by Sahami et al. [16], use the conditional probability of tokens occuring in spam and legitimate messages to distinguish between these two types of messages. The evaluation by Androutsopoulos et al. [1] showed that these approaches are viable but not without shortcomings. The advantages of these approaches is that they are user-specific and offer low false positive and false negative rates after sufficient training with the current generation of spam. Their disadvantages are that this training process is rather time-consuming, and the resulting training statistics cannot be easily re-used or combined for different users. Additionally, as mentioned above, spammers are able to evade these approaches by using common words [21] and camouflage [20] to augment their spam messages.

Exchange Intelligent Filter [4] and SpamAssassin [18] are two examples of content-based approaches that use heuristics to filter spam. Both tools calculate a score for every message based on manually extracted sets of features (rule bases). The disadvantage of these heuristic-based methods is that they are very ad hoc and require regular updates to ensure accuracy. Additionally, these updates can be as complex as the filters themselves [7]. The current version of SpamAssassin attempts to deal with this problem by including a number of plugins that support different methods (including Bayesian filtering) to improve the score calculation. One of these plugins is related to our approach: SURBL/SpamCopUri [2]. This plugin blocks messages by using a blacklist of URLs. The blacklist is created based on the spam submissions received from users. One disadvantage of this method is that it takes time for spam to be reported, and by the time an update is received, it is already too late (i.e., users have already received the spam). Also, it is very easy for spammers to change the text of a URL and have it point to the same content (e.g., a redirect).

Our approach is different from SURBL in three ways. First, the definition of spam in our approach is personalized. Each user has a different set of categories which correspond to that particular user's interests. Additionally, this information can be combined and shared easily among users. Second, in our system the presence of a URL in a message is a good indicator of spam. Third, in addition to the text of a URL, our approach may also use the information from the website referenced by that URL. For similar reasons, our approach is also different from the URL module used by Brightmail [17].

3 Description

As spammers become more sophisticated, the token signatures (e.g., tokens found in the message body) used by text-based filters to distinguish between spam and legitimate messages will no longer be valid. Thus, we must focus our attention on the characteristics of spam messages which spammers are unable to successfully obscure. A very clear example of such a characteristic is the URLs found in spam messages. Spammers rely on these URLs as a feedback mechanism, and as a result, the URLs must be accessible to the users. This required accessibility introduces a new technique for filtering spam messages.

In our new approach, we filter email messages based on the URLs they contain. If the URLs in a particular message point to websites which are of interest to the user, then that message is considered legitimate; otherwise, the message is considered spam. We determine a user's interest in a URL (and its corresponding website) using a number of techniques: URL category whitelists, URL regular expression whitelists, and dynamic classification of websites. In the following sections, each of these techniques is described in more detail.

3.1 URL Category Whitelists

Many search engines (e.g., Google, Yahoo!, LookSmart, etc.) maintain directories that contain category information for websites. For example, Google's directory [8] categorizes http://www.google.com as Computers/Internet/Searching/Search Engines/Google. Using these directories, we are able to categorize the URLs a user is interested in, compiling a list of acceptable categories $A_{categories}$. Then, we can use $A_{categories}$ to classify incoming messages as either legitimate or spam based on the URLs they contain. When a new message arrives, the URLs found in that message are categorized. If all of the corresponding categories match categories in $A_{categories}$, then the message is classified as legitimate. Otherwise, the message is classified as spam. Unfortunately, not all URLs are listed in the search engines' directories. For the remainder of this paper, we will use the term *uncategorized* URLs to refer to these URLs which are not listed in any of the directories, and we will use the term *categorized* URLs to refer to URLs which are listed in at least one of the directories. In the next section, we describe an additional technique employed to help handle uncategorized URLs.

3.2 URL Regular Expression Whitelists

Using search engine directories to categorize and classify URLs is a novel solution, but it is not always successful. As previously mentioned, uncategorized URLs are not listed in these directories. Additionally, in some cases, users have an interest in websites that cannot be expressed easily with categories. For example, a user might want to register an interest in all websites under a given top-level domain (e.g., .edu, .mil, etc.). For these special cases, an additional technique is needed to classify the URLs. One possible technique is the construction of a URL regular expression whitelist which contains a list of acceptable regular expressions A_{regex}. When a new message arrives, the URLs found in that message are compared to the regular expressions in A_{regex}. If all of the URLs match at least one of those regular expressions, then the message is classified as legitimate. Otherwise, the system obtains the categories for the URLs that did not match any of the regular expressions and compares those categories to $A_{categories}$ (as explained in the previous section). Unfortunately, this process might still result in uncategorized URLs. Thus, in the next section, we explain another approach used to deal with the remaining uncategorized URLs.

3.3 Website Classification

In addition to creating the two types of whitelists described above, our system also retrieves the contents of the websites referenced by the URLs used to create $A_{categories}$ and A_{regex}. The system uses these website contents to train a learning spam filter (e.g., Naive Bayes, Support Vector Machines, LogitBoost, etc.) which is used to classify uncategorized URLs that do not match any of the regular expressions in A_{regex}. When a new message arrives containing these types of URLs, the system retrieves the contents of the websites referenced by those URLs. Then, the learning spam filter is used to classify each of the websites. If the filter classifies one of the websites in the message as spam, then that message is classified as spam. Otherwise, if all of the websites in the message are classified as legitimate, then that message is classified as legitimate.

4 Implementation

Our system can be implemented as either server-based or client-based. In the server-based implementation, all users' mail is filtered by a central server, and that server keeps track of each user's profile. In the client-based implementation, each client runs a separate copy of our system. The main advantage of the server-based implementation is the improved performance obtained by maintaining a global cache of URL information (see Section 4.1 for a description of this cache). The main advantage of the client-based implementation is the improved privacy protection it provides each user.

Our implementation is server-based. It uses a RedHat Linux v9 machine which runs an Apache web server with `mod_ssl`. Our actual system consists of two parts: the web-based configuration interface and the mail classifier. The mail classifier is implemented using Perl and procmail, and it also includes a learning spam filter based on POPFile [9]. The system's operation is broken into two phases: training and configuration. These two phases are described in detail in the following sections.

4.1 Training

Since each user has unique website interests, a separate profile is maintained for every user in the system. Each of these profiles consists of three main parts:

- A list of acceptable categories $A_{categories}$ (as explained above in Section 3.1).
- A list of regular expressions A_{regex} for acceptable URLs (as explained above in Section 3.2).
- A trained learning spam filter (as explained above in Section 3.3).

A user's profile is created during the system's training phase. By default, this profile is generated automatically by the system, but the user also has the option of creating the profile manually. After the automatic profile generation, users may also edit and verify $A_{categories}$ and A_{regex} to ensure their interests are properly reflected. Additionally, our system provides three pre-defined profiles for academic, business, and home users. These profiles can be used without modification; they can serve as a template for users, or they can be disregarded completely. A partial example of the Academic profile is given in the Appendix.

The automatic profile generation process occurs as follows. First, training URLs are extracted from the user's existing legitimate email messages. Additional URLs can also be obtained from the user's Bookmarks/Favorites list which is maintained by the user's favorite web browser. Once the training URLs are obtained, they are categorized using the directories of multiple search engines (e.g., Google, Yahoo!, LookSmart, etc.), and the corresponding categories are stored in $A_{categories}$.

Since it may take several seconds to query the search engines for each URL, our system also maintains a system-wide cache for query results to improve performance. This cache contains category and other information retrieved from search engines, and its entries are periodically expired to ensure the information remains current. Thus, when a user needs to query the search engines, the cache is consulted first. If the necessary information is not found there, then the query is forwarded to the actual search engines.

After the categories are stored in $A_{categories}$ and the system-wide cache, regular expressions are created to match each of the user's unique, uncategorized training URLs. These regular expressions are then stored in A_{regex}. Next, the system retrieves the contents of the websites referenced by the URLs used to create $A_{categories}$ and A_{regex}. The system also retrieves the contents of the websites referenced by the URLs present in the user's existing spam messages. Once the system has these websites' contents, those contents are used to train the system's learning spam filter.

An immediate problem that arises when obtaining a website's contents is redirection. Spammers can easily use multiple redirects to hide their real website. Thus, any attempt to obtain website content must handle redirects correctly. In our system, we resolve this issue by relying on the cached copies of websites which are stored by search engines. When the search engines index websites, their crawlers automatically follow the redirects. Thus, the cached copies stored by the search engines are the end-sites rather than the redirecting pages. Our system's learning spam filter uses these cached copies during its training and classification phases.

4.2 Classification

Once the training phase is complete, the system uses the user's profile to classify incoming email messages. This classification process works as follows. When a new message arrives, it is scanned for URLs. If the message does not contain any URLs, it is classified as legitimate. Otherwise, the system checks every URL in the message according to the following process.

First, each URL is compared to the regular expressions in A_{regex}. If all of the URLs match at least one of those regular expressions, then the message is classified as legitimate. Otherwise, the category information is obtained for the URLs that did not match any of the regular expressions. To improve performance and reduce the load placed on the search engines, the system initially consults the system-wide cache for each URL's category. If the cache does not contain the necessary information, then the search engines are queried, and the results are placed in the cache. Once the system has the categories for the URLs, those categories are compared to the categories in $A_{categories}$. For every URL with a category found in $A_{categories}$, a new regular expression is created and added to A_{regex}. The purpose of this new regular expression is to optimize the system's performance when this URL is encountered again, and we refer to this optimization process as *incremental learning*. If all of the URLs have categories found in $A_{categories}$, then the message is classified as legitimate. Otherwise, if one of the URLs has a category not found in $A_{categories}$, the message is classified as spam. However, if some of the URLs are uncategorized, then the learning spam filter is used to classify the contents of the websites referenced by those URLs. If all of those websites are classified as legitimate, then the message is classified as legitimate. Otherwise, the message is classified as spam.

5 Experimental Evaluation

To validate our new approach to spam filtering, we conducted a couple preliminary experiments comparing SpamAssassin v.2.63 [18] and our system on two email message corpora ($C_{legitimate}$ and C_{spam}). $C_{legitimate}$ consisted of 3560 manually filtered legitimate messages which were received by one of the authors over a period of two years. C_{spam} consisted of 1496 recent spam messages which were obtained

from the publicly available corpora maintained by SpamArchive[1]. Each of our experiments were conducted as if a single user were using SpamAssassin and our system.

5.1 Training

For the training sets in our experiments, we used the first 1191 and 493 messages from $C_{legitimate}$ and C_{spam}, respectively. SpamAssassin was trained with these training sets using its text-based filter, `sa-learn`. Our system was trained according to the process described in Section 4.1. During this training process, the system identified 230 unique categories and a number of uncategorized URLs.

We found that the list of uncategorized URLs included many duplicates, and most of these duplicates were what we refer to as *footer* URLs. These footer URLs included the URLs added by free webmail services (e.g., Yahoo! and Hotmail), links to personal websites from friends and co-workers, and URLs added by automatic mail filtering tools (e.g., Anomy Sanitizer [6]).

5.2 Classification

After both SpamAssassin and our system were trained, we evaluated their performance by using them to classify our legitimate and spam workloads. For these workloads, we used the remaining 2369 and 1003 messages from $C_{legitimate}$ and C_{spam}, respectively, that were not included in the training sets. In the following sections, we compare the false positive and false negative rates for SpamAssassin and our system.

5.2.1 Legitimate Workload

SpamAssassin classified 56 of the 2369 legitimate messages as spam (2% false positive rate). Using this result as a baseline, we performed three experiments to test various aspects of our system's performance. First, we evaluated the success of only using URL categories to classify messages. Then, we investigated the effect of using regular expressions to classify uncategorized URLs. Finally, we studied the effectiveness of using a Bayesian spam filter to classify the websites referenced by the uncategorized URLs which were not handled by the previous two experiments.

In our first experiment, we only used URL categories to classify the messages in the workload. The purpose of this experiment was to justify our proposed technique of message classification using the category information maintained by search engines. Thus, for this experiment, we only used 1373 of the 2369 messages (57%) because they exclusively contained categorized URLs (i.e., the messages did not contain any uncategorized URLs) or no URLs at all. For each of these messages, our system extracted the message's URLs and obtained the categories corresponding to those URLs. Then, these categories were used to classify the message using the process described in Section 3.1. Using that process, only 14 of the 1373 messages were classified as spam (1.02% false positive rate).

This experiment illustrates two important points. First, email messages which exclusively contain categorized URLs can be classified very effectively using category information. Second, using only category information is insufficient for all email messages because many messages contain uncategorized URLs. Our next two experiments evaluate techniques we developed to handle these uncategorized URLs.

The second experiment we conducted was an extension of the first experiment. Our system still used URL categories to classify messages, but it also used regular expressions to help classify messages containing uncategorized URLs. As a result, for this experiment, the system was able to classify all of the messages

[1] SpamArchive's spam corpora can be found at ftp://spamarchive.org/pub/archives/.

in the workload. The classification process was the same as the one described in Section 3.2. Using that process, 198 of the 2369 messages were classified as spam (8.36% false positive rate). 33 of these messages were misclassified based on category information, and 165 were misclassified because they contained uncategorized URLs that did not match any of the URL regular expressions. The number of category-related misclassifications is higher than in the previous experiment because that experiment only dealt with messages which contained exclusively categorized URLs or no URLs at all. In this experiment, messages containing both categorized and uncategorized URLs were used. As a result, more messages were classified based on the categories of the URLs they contained.

This experiment shows that our system can be used to classify messages that contain both categorized and uncategorized URLs. However, the false positive results for this experiment are not as good as those produced by SpamAssassin. Thus, in our next experiment, we tested a method of dealing with the uncategorized URLs which did not match the regular expressions.

The results of our first two experiments were very encouraging; however, in those experiments, our system was unable to handle a number of uncategorized URLs. To address this situation, our third experiment combined the processes used in the previous two experiments, and it also employed a Bayesian spam filter to classify the websites referenced by those uncategorized URLs which the previous two experiments were unable to handle. The process used to classify these websites was described in Section 3.3. By combining all of these techniques, our system only classified 40 of the 2369 messages as spam (1.69% false positive rate). By comparing this value to the false positive rate obtained with SpamAssassin, we find that our system was actually more effective at classifying messages in the legitimate workload than SpamAssassin. Obviously, our system's performance will vary based on each user's individual profile, but these preliminary results are still quite encouraging. In the next section, we present the results we obtained by using both SpamAssassin and our system to classify the spam workload.

5.2.2 Spam Workload

SpamAssassin correctly classified all 1496 messages in C_{spam} as spam (0% false negative rate). Only 1313 of the 1496 messages (88%) in C_{spam} contained at least one URL, and our system was able to classify all but 1 of those 1313 messages as spam (0.076% false negative rate). Thus, our system was very effective at identifying spam messages containing URLs.

Note that the percentage of messages in C_{spam} containing at least one URL is lower than the percentages cited by other researchers (e.g., more than 95% according to [17]). Manual analysis of the messages in C_{spam} revealed that many of the messages without URLs were already partially filtered or anonymized (either by SpamArchive or the users which submitted the messages to SpamArchive). Additionally, a number of these messages were meaningless junk email submissions with no contact information. Due to these observations, we believe the percentage of spam messages "in the wild" which contain URLs is closer to the 95% value cited in [17]. Consequently, our new approach to spam filtering is very successful for a vast majority of spam messages. However, for messages that do not contain URLs, an additional approach is required.

5.2.3 Summary

Using the three techniques discussed in Section 3, our system was able to attain a false positive rate of 1.69%. By comparing this rate to SpamAssassin's false positive rate on the same workload (2%), we find that our system is very effective when identifying legitimate messages. Unfortunately, our approach only considers messages containing URLs; thus, it has a higher false negative rate than SpamAssassin. However,

overall, this new approach to spam filtering is still extremely effective, and it concretely illustrates the benefits associated with URL-based spam filtering.

6 Conclusions

In this paper, we presented a new method of filtering spam that focuses on the presence of URLs as spam signatures. URLs are very reliable indicators of spam since they need to be "live" in order for spammers to profit from potential contact by victims. This method complements the current generation of token-based spam filters which are vulnerable to spammers' manipulation of spam message content.

First, we parse the messages and identify the URLs they contain. Then, we use URL category information already maintained by search engines to check the validity and content classification of those URLs. We show that URL-based spam filtering is viable and that it can significantly improve the overall effectiveness of spam filtering. Our results suggest that further exploration of such spam filtering methods is warranted.

Appendix

Academic Profile: Category Whitelist

```
Science/Conferences
Science/News
Science/Publications
Computers/Computer_Science/Organizations
Computers/Computer_Science/Research_Institutes
Computers/Computer_Science/Academic_Departments/North_America/United_States/Georgia
Science/Technology/Academia
News/Colleges_and_Universities/
Computers/Internet/Policy
Computers/Internet/Searching/Search_Engines/Google
Computers/Supercomputing
Computers/Systems/
News/By_Subject/Information_Technology
News/By_Subject/Information_Technology/Computers
News/By_Subject/Information_Technology/Internet/Headlines_and_Snippets
...
```

Academic Profile: Regular Expression Whitelist

```
*.edu
*.gov
*.org
*.mil
*.yahoo.com
*.google.com
*.cos.com
citeseer.ist.psu.edu
www.research.ibm.com
www.research.att.com
www.research.microsoft.com
www.nytimes.com
www.cnn.com
...
```

References

[1] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000)*, pages 160–167, Athens, Greece, July 2000.

[2] J. Chan. Surbl - spam uri realtime blocklists. http://www.surbl.org/, 2004.

[3] W. W. Cohen. Learning rules that classify e-mail. In *1996 AAAI Spring Symposium on Machine Learning and Information Access*, Palo Alto, CA, 1996.

[4] Microsoft Corporation. Exchange intelligent message filter. http://www.microsoft.com/exchange/downloads/2003/imf/default.mspx, 2003.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, chapter Bayes Decision Theory, pages 10–43. Wiley, New York, New York, 1973.

[6] B. R. Einarsson. The anomy mail tools. http://mailtools.anomy.net/, 2004.

[7] J. Goodman. Spam filtering: From the lab to the real world. MIT Spam Conference, 2003.

[8] Google. Google directory. http://dir.google.com/, 2005.

[9] J. Graham-Cumming. Popfile - automatic email classification. http://popfile.sourceforge.net/, 2004.

[10] E. Harris. The next step in the spam control war: Greylisting. http://projects.puremagic.com/greylisting/, 2003.

[11] M. Iwanaga, T. Tabata, and K. Sakurai. Evaluation of anti-spam method combining bayesian filtering and strong challenge and response. In *IASTED International Conference on Communication, Network, and Information Security (CNIS 2003)*, pages 214–219, 2003.

[12] E. Kidd. Bayesian whitelisting: Finding the good mail among the spam. http://www.randomhacks.net/stories/bayesian-whitelisting.html, 2002.

[13] R. E. Kraut, S. Sunder, J. Morris, R. Telang, D. Filer, and M. Cronin. Markets for attention: Will postage for email help? In *2002 ACM conference on Computer supported Cooperative Work*, pages 206–215, New Orleans, LA, 2002.

[14] M. Lamb. Tarproxy: Lessons learned and what's ahead. MIT Spam Conference, 2004.

[15] V. V. Prakash. Vipul's razor. http://razor.sourceforge.net/, 2004.

[16] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI Workshop on Learning for Text Categorization*, pages 55–62, Madison, Wisconsin, July 1998.

[17] K. Schneider. Brightmail url filtering. MIT Spam Conference, 2004.

[18] SpamAssassin Development Team. The apache spamassassin project. http://spamassassin.apache.org/, 2004.

[19] B. Templeton. E-stamps. http://www.templetons.com/brad/spam/estamps.html, 2003.

[20] S. R. Webb, S. Chitti, and C. Pu. An experimental evaluation of spam filter performance and robustness against attack. submitted for publication in CEAS 2005, 2005.

[21] G. L. Wittel and S. F. Wu. On attacking statistical spam filters. In *1st Conference on Email and Anti-Spam (CEAS 2004)*, 2004.