On Attacking Statistical Spam Filters

Gregory L. Wittel and S. Felix Wu

Department of Computer Science
University of California, Davis
One Shields Avenue, Davis, CA 95616 USA

Abstract. The efforts of anti-spammers and spammers has often been described as an arms race. As we devise new ways to stem the flood of bulk mail, spammers respond by working their way around the new mechanisms. Their attempts to bypass spam filters illustrates this struggle. Spammers have tried many things from using HTML layout tricks, letter substitution, to adding random data. While at times their attacks are clever, they have yet to work strongly against the statistical nature that drives many filtering systems. The challenges in successfully developing such an attack are great as the variety of filtering systems makes it less likely that a single attack can work against all of them. Here, we examine the general attack methods spammers use, along with challenges faced by developers and spammers. We also demonstrate an attack that, while easy to implement, attempts to more strongly work against the statistical nature behind filters.

1 Introduction

As the volume of unsolicited bulk e-mail increases, it is becoming increasingly important to apply techniques that mitigate the cost of spam. With the increasing popularity of anti-spam measures, spammers have been forced to find new ways to ensure delivery of their messages. Their responses include changing message content, increasing message volume, new delivery mechanisms, and attacking anti-spam groups. As such, spam filtering has become an arms race with spammers and anti-spammers countering each other's tactics. Spammers' attacks are generally simple and do not work against the underlying principles of statistical filters. Instead, their attacks manipulate basic mechanisms such as hashing, parsing, and tokenization. In an effort to strengthen existing systems, it is useful to examine current evasion methods as well as considering other potential attacks against spam filters.

A basic introduction to spam filtering is given in Section 2.1. This section can be skipped by those who already have an understanding of how text classification systems work. The remainder of Section 2 covers some of the related work in attacking spam filters. Various types of attacks against spam filters are summarized in Section 3. Challenges faced by developers and spammers are also discussed. Section 3 concludes by considering an attack methodology and testing a refined version of a dictionary attack against two filters. The last section, Section 4, summarizes this paper and talks about possible future directions in looking at attacks against filters.

2 Background

2.1 Filtering Overview

Over the past few years, spam filtering software has gained popularity due to its relative accuracy and ease of deployment. With its roots in text classification research, spam filtering software seeks to answer the question "Is message x spam?". The means by which this question is addressed varies upon the type of classification algorithm in place. While the categorization method differs between statistical filters, their basic functionality is similar. The basic model is often known as the bag of words (multinomial) or multivariate model [6]. Essentially, a document is distilled into a set of features such as words, phrases, meta-data, etc. This set of features can then be represented as a vector whose components are boolean (multivariate) or real values (multinomial). One should note that with this model the ordering of features is ignored.

From here, the classification algorithm uses the feature vector as a basis upon which the document is judged. The usage of the feature vector varies between classification methods. Two of the most common methods used in spam filtering are rule based and statistics driven. As the name implies, rule based methods classify documents based on whether or not they meet a particular set of criteria. Rules may be hand-crafted or automatically generated. Machine learning algorithms are primarily driven by the statistics (e.g. word frequency) that can be derived from the feature vectors. One of the widely used methods, Bayesian classification, attempts to calculate the probability that a message is spam based upon previous feature frequencies in spam and legitimate e-mail [9, 8, 1, 4]. Other notable learning algorithms applied to spam filtering include boosting [2] and support vector machines [10, 3].

2.2 Related Work

Relatively little work studying attacks against spam filters has been done. Instead, efforts appear to have been reactive as developers adjust to different tactics used by spammers. Recently, John Graham-Cumming studied one type of stronger statistical attack [5]. Expanding on the concept of adding random words to a given spam, his attack focused on the configuration of a particular filter. By doing so he was able to isolate particular words that when included in a spam would cause the message to be marked as ham (non-spam e-mail). Due to the effort involved his attack would not be practical to target a single user, but it might be useful when working against a large group sharing a single filter configuration.

3 Breaking Filters

3.1 Attack types

As filtering has become more prevalent, spammers have started modifying their messages in an attempt to bypass filters. Many of the methods that spammers use have been cataloged in the Spammers' Compendium[1]. The evasion techniques that spammers use can be roughly grouped into several categories:

Tokenization With this attack, the spammer is working against the feature selection (tokenization) of a message by splitting or modifying key message features. Examples include splitting up words with spaces, and using HTML (or Javascript/CSS) layout tricks.

Obfuscation Here, the message's contents are obscured from the filter using encoding or misdirection. This would include HTML entity/URL encoding, letter substitution, Base64/UUencode/Quoted printable encoding, etc.

Weak Statistical Such attacks attempt to skew the message's statistics such that a filter has trouble telling if a message is spam or not. An attack is termed weak if its approach uses purely randomized data. Using random words, fake HTML tags, or random text excerpts are common versions of such attacks.

Strong Statistical A strong statistical attack is differentiated from a weak one by the nature of the data added. One can think of a weak attack as guessing what might distract the filter whereas a strong one makes an educated guess. By using more focused 'random' data, the chances of successfully circumnavigating a filter may be higher. However, the difficulty in developing attacks is also increased. Because of this, the practicality of a strong attack is limited (from a spammer's point of view). A statistical attack can also be strengthened by taking advantage of feedback methods to record which spams made it to end users. Examples include the attack demonstrated by Graham-Cumming [5].

Other attacks do not quite fit in the above groupings. These include sparse data attacks (spams with only a few words or URLs), and hash breaking attempts (adding in random characters).

[1] http://www.jgc.org/tsc/

3.2 Challenges

Developers face a number of problems in building an accurate and resilient filtering system. Dealing with attacks further complicates filter implementation. Over time spam changes as bulk mailers attempt to bypass filtering systems. This aspect forces developers to continually test against and adapt to varying traits of spam. When testing a filter system, the selection of a test corpus poses some issues. Factors to consider in a corpus include the data's age, breadth (variety of sources), accuracy, and size. Age is important as the content of spam changes over time. Sampling corpus data from a number of sources is necessary in order to get a broad view of the types of messages being sent. Also, there are many non-spams present in the SpamArchive.org collection[2]. Should such erroneous classifications be used to test/train filters, developers will have inaccurate information on a filter's performance. While there are a number of public spam collections, few public ham datasets are available. Privacy issues aside, it is difficult to build a ham corpus that is representative of differing end-user profiles. End users also provide a challenge to filter developers. Having filters tailored to each user's usage patterns will yield better results, but it is easier to deploy system-wide filters. End-users marking legitimate messages as spam becomes problematic in an organization wide configuration.

Spammers also have a number of obstacles in developing attacks against a filter. In attempting to design messages that are able to pass through filters undetected, a spammer needs to preserve as much of the intended message as possible. Additionally, the spammer would prefer to maximize audience size while keeping effort minimal. The challenge lay in identifying the configuration of a target. If the system is learner based, the vulnerabilities are dependent on the dataset the filter uses to make decisions. With such goals in mind, it is easier to attack a mono-culture of filters (e.g. organizational deployments). Together, these issues make building sophisticated attacks against filters a hard problem for a spammer.

3.3 Attack Methodology

A major advantage filter developers have over spammers is that most of the content resulting from evasion methods are unique to spam. For this reason, most of the simpler attacks do not work. The question becomes, how would a spammer go about developing an advanced attack? An attack method can be divided into wants and capabilities. In addition to balancing effort and gain, a spammer would like an attack to be repeatable. In order to achieve this goal, a spammer must rely on either luck or strength of attack. Luck in the sense that the attack (message) is missed by the filter or is not used to retrain the filter. Strength of attack when subsequent attempts cannot be detected by the filter despite retraining on earlier instances. Along with repeatable attacks, a spammer would like a way to get feedback in order to tell which messages made it to the end user. The embedding of web bugs can perform this task assuming the spammer is able to provide a stable way to host the bugs. The viewing of a web bug only guarantees that the message was accessed by something. It does not supply the spammer with knowledge that the message was viewed by an end user or left unmarked by filters.

In terms of attack related capabilities, an adversary can study the target filters from two angles. By using an inside account he can test and refine attacks. With an inside account one can study the target filtering system at the organizational level. While using various anti-spam tools the spammer can look for implementation weaknesses in addition to testing his attacks. A spammer also has a few different delivery methods available. Common methods include sending directly to the target system, using open relays, or using a proxy to perform the message transmission (e.g. MyDoom infected machines). Other capabilities include varying the primary spam message content/style and URLs referenced.

3.4 Example attack and results

The goal in attacking a filter is to have an automated way of transform a given (spam) message into a stealthy spam. The most effective method depends on the target filter type and configuration. An attack where one appends random words to a short spam (a picospam) was previously seen to be ineffective [5]. The attack failed because random dictionary words are not likely to belong to the set of "good" words.

[2] e.g. Messages 4-527, 60-1549 (mailboxnumber-msgnumber); ftp.spamarchive.org/pub/archives/submit/

```
1  From: Kelsey Stone <bouhooh@entitlement.com>
2  Subject: Erase hidden Spies or Trojan Horses from your computer
3
4  Erase E-Spyware from your computer
5
6  http://boozofoof.spywiper.biz
```

Fig. 1. Abbreviated source picospam. (Message 24-3379 from SpamArchive.)

However, the basic attack can be refined to choose the random words from a list of common English words. In theory, this should increase the chances that the random words will appear in the list of desirable ones; thus increasing the attack's strength. As a base dataset, 3000 spams and 3000 hams were randomly selected from the SpamArchive.org and SpamAssassin[3] corpora respectively. The attacks were tested against SpamBayes version 1.0a9 and CRM114 release 20040312. SpamBayes was trained on the selected messages, while CRM114 was trained only when it made classification errors (as recommended by the documentation).

Starting with a picospam that is detectable by the tested filters (*Fig. 1*), n random words were added to the message. For each n, this was repeated 1000 times. The random words were chosen with and without replacement (allowing repeats v. no repeats). This resulted in two datasets for $n = 10, 25, 50, 100, 200, 300, 400$; totaling 7,000 messages per set. The generated spam variations were then processed by each filter and the message scores with number of false negatives was recorded. The above tests were performed twice: First using words from a dictionary (as a baseline), and second with a list of common English words [7] (slightly modified by removing spammy words). In all cases, the results between attacks with and without word replacement did not significantly vary. Thus, all results reported are for attacks with replacement word selection.

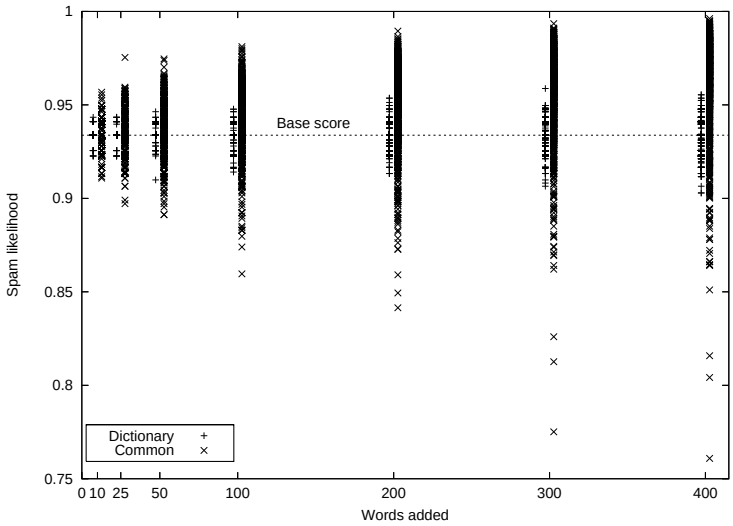


Fig. 2. CRM114 - Spam score on dictionary and common word attacks.

With the dictionary attacks CRM114 performed extremely well with zero false negatives. Using attack words from a dictionary, SpamBayes' performance was not so good (*Fig. 3(a)*). This result is surprising as the dictionary attack's level of success contradicts what was found by Graham-Cumming for this type of

[3] SpamArchive: http://www.spamarchive.org/, SpamAssassin: http://spamassassin.org/publiccorpus/

Words	Spam	Ham	Unsure
10	999 / 1000	0 / 0	1 / 0
25	937 / 772	0 / 0	63 / 228
50	484 / 16	0 / 0	516 / 984
100	22 / 0	0 / 943	978 / 57
200	0 / 0	269 / 1000	731 / 0
300	0 / 0	829 / 1000	171 / 0
400	0 / 0	858 / 1000	142 / 0

(a) Before training.

Words	Spam	Ham	Unsure
10	1000 / 1000	0 / 0	0 / 0
25	967 / 872	0 / 0	33 / 125
50	601 / 56	0 / 0	399 / 944
100	37 / 0	0 / 735	963 / 265
200	0 / 0	78 / 1000	922 / 0
300	0 / 0	625 / 1000	375 / 0
400	0 / 0	695 / 1000	305 / 0

(b) After training on source picospam.

Fig. 3. SpamBayes - Classifications using dictionary/common word attack. Each cell has two numbers reporting classification counts: the first refers to the number of dictionary attack messages and the second to the common word attack.

attack [5]. This difference in results may be due to differences between training corpora or how the tested filters score messages.

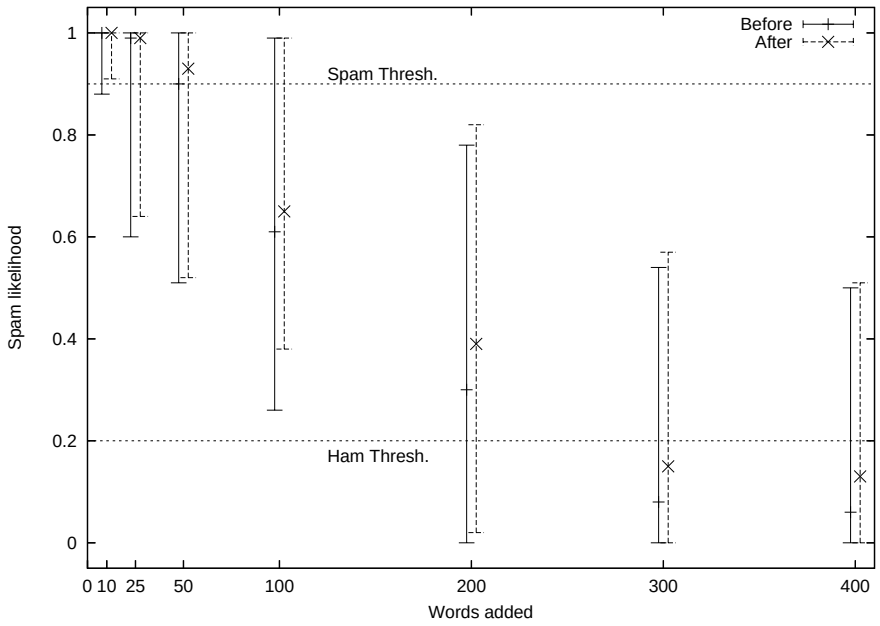
Because CRM114 performed so well against the dictionary word attack, its resistance to the common word attack is not surprising (zero false negatives). What is interesting is that the dictionary attack score spread is much more confined than that of the common word attack (*Fig. 2*). The bulk of the common word attack instances seem to cause a greater shift toward a stronger spam score (the median score increased linearly). This may be caused by the presence of sequences deemed 'bad' by CRM114 in the common word attacks causing a large shift in the score. On the other hand, SpamBayes performed worse against the common word attack with 100 attack words being the major transition point. With as few as 50 common words added, there is a significant shift toward being unable to identify the messages as spam or ham.

One possible explanation is that SpamBayes had not been trained the basis picospam (despite easily detecting it without being trained on the message). To test this hypothesis, the original picospam was added to SpamBayes' training data. After retesting, SpamBayes performed better against both attacks but it appears to still be vulnerable to the common word attack (*Fig. 3(b)*). Figure 4 shows the range of message scores assigned to the attack variants for the cases before and after training on the original picospam. For comparison, the picospam without any attack data received a spam probability of 1.0 regardless of training. The bars indicate the high, low, and median score for each group. When classifying messages, SpamBayes internally generates two scores, 'H' and 'S', from which the spam probability score is calculated. Under regular conditions, a spam message should result in a high 'S' and a low 'H' score while ham messages do the opposite. Looking at the filter's behavior under attack conditions, we saw that the 'S' and 'H' values shifted from high/low toward low/high as the attack size increased. Essentially, the attacks cause SpamBayes to do exactly what it was designed to do: classify messages with lots of "good" features and relatively few "bad" features as legitimate.

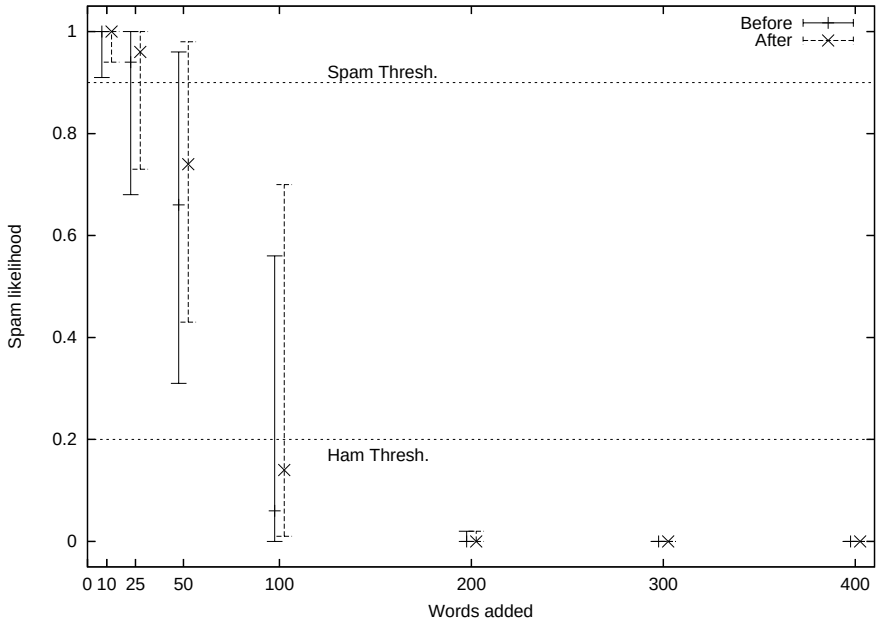
Overall, this attack has shown mixed results. While it affected the message scoring, the shift was not significant enough to cause errors with CRM114. With SpamBayes, the common word attack quartered the number of words needed to 'break' SpamBayes compared to a dictionary attack. Training on the source picospam only slowed the transition from 'unsure' to 'ham' slightly. Though these tests show that it is possible to influence the filter's result, a larger scale test is needed in order to check for statistical significance and if the attack works against other types of filters.

4 Conclusion

Most attacks by spammers do not strongly attack the statistical base of spam filters. Instead, they primarily work by attacking the filter's feature generation through tokenization or obfuscation. When developing a filter, programmers have a number of attack and general issues that must be considered. Spammers face a



(a) Dictionary word attack



(b) Common word attack

Fig. 4. SpamBayes - Comparison of performance before and after trained on base picospam. The bars indicate the high, low, and median scores for the tested attacks.

greater challenge in attempting to defeat filters. Their task is akin to attempting to learn how to coax a black box into producing a specific output. Whether or not stronger attacks against filters are adopted is tied to the ubiquity (and effectiveness) of spam filtering systems. Spammers will only adopt new methods against filters if the gain is perceived as greater than the required effort. Though the common word attack showed it worked much better than the dictionary attack (requiring one fourth as many attack words), it also only worked against one of the two tested filters. Thus, in addition to looking at more test data, it would be useful to compare how a larger pool of filters handle the attack. In order to determine factors that result in vulnerability to this attack, a test controlling various parameters such as feature selection and learning algorithms is necessary. The effect of retraining against attack messages needs to be studied as well. In particular, looking at the effect retraining using common word attack messages has on false positive rates.

Looking beyond advanced statistical attacks, it may be useful to consider traditional security flaws in filtering software. One example is a buffer overflow vulnerability found in a component of Symantec's Norton AntiSpam software[4]. The format of e-mail as required by various RFCs (i.e. 2822, 2045) may limit the scope of such attacks. However, as with the Symantec example the exploit code itself need only be referenced by the message and not directly included. Another area of interest is the application of natural language processing/generation to spams. Using a limited vocabulary of 'good' words, a message may be generated or transformed into one that appears less suspicious to filters. Regardless of the presence of new attacks, the struggle between spammers and anti-spammers has no clear end in sight. With any anti-spam measure it is important to also consider for whom the cost of spam is reduced. Though spam filtering works well, filtering alone cannot limit all of such costs associated with spam.

References

1. Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age*, pages 9–17, 2000.
2. Xavier Carreras and Lluís Màrquez. Boosting trees for anti-spam email filtering. In *Proceedings of the 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BG, 2001.
3. Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10:1048–1054, 1999.
4. Paul Graham. A plan for spam. WWW, August 2002. http://www.paulgraham.com/spam.html.
5. John Graham-Cumming. How to beat an adaptive spam filter. MIT Spam Conference, 2004.
6. Peter Jackson and Isabelle Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization.* John Benjamins Publishing Company, 2002.
7. Wortschatz Lexicon. Wortlisten. WWW, August 2001. Lists of common words in German, English, Dutch, and French. http://wortschatz.uni-leipzig.de/html/wliste.html.
8. Patrick Pantel and Dekang Lin. Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, pages 95–98, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
9. Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, pages 55–62, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
10. Donghui Wu and Vladimir Vapnik. Support vector machine for text categorization, 1998.

[4] See advisory at: http://www.securityfocus.com/archive/1/357954