International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 2, Issue 4 (July-Aug 2014), PP. 153-156

DEVELOPMENT OF AN EFFECTIVE BAYESIAN APPROACH FOR SPAM FILTERING

Pranish Shukla[1], Manoj Kumar[2]
M.Tech,BBD University, Lucknow
Assistant Prof. IT Department, BBDNITM, Lucknow

ABSTRACT: Email spam, also known as junk email or unsolicited bulk email(UBE), is a subset of electronic spam involving nearly identical messages sent to numerous recipients by email. Clicking on links in spam email may send users to phishing web sites or sites that are hosting malware. Spam email may also include malware as scripts or other executable file attachments. Definitions of spam usually include the aspects that email is unsolicited and sent in bulk

In order to overcome spam problem many researchers have been conducted and various method of anti-spam filtering have been implemented. A spam filter is a set of instruction for determining the status of the received email. Spam filters are used to prevent spam email passing through the recipient. The main challenge is how to design an effective spam filter that allows desired email to pass through while blocking the unwanted email.

I. INTRODUCTION:

The evaluation of anti-spam solution is measured by its effectiveness, accuracy and ease of administration. Accuracy refers to the percentage of spam messages correctly filtered as spam. To maintain high effectiveness for an anti-spam solution is challenging for a number of reasons. Firstly, the spam that constitutes the final few percentage points that can often be quite subjective and difficult to address on a global, server-wide basis. Client-side filters may be an option in such cases. Next, an over-aggressive drive towards effectiveness increases the chances of legitimate mail being blocked. Finally, as spam-blocking percentages increases, spammers have an incentive to increase their overall volume to maintain the status quota. The adaptability of spammers, today‘s defenses can be obsolete as spammers find workarounds in near future. To avoid such consequences vendors must continuously monitor and tune their filters from time to time. Evaluating current effectiveness the vendors should demonstrate a commitment and infrastructure support necessary to keep pace with spammers. Another integral criteria for an anti-spam solution is its accuracy. Effectiveness refers to the ability of the anti-spam software to distinguish legitimate messages from spam. As false positives increase, recipients lose trust in spam blocking techniques. When users feel they can‘t rely on these anti-spam filtering techniques, they are forced to manually wade through their quarantines and delete spam, a risky and frustrating exercise. An anti-spam solution should be 100% effective and 100% accurate. To block enough spam, the solution must be aggressive.

II. False negative

If you receive a message that passed through the spam filters that you feel should be classified as spam, you can submit this false negative message, who will review the message and add it to the service-wide spam filters if it meets the spam classification criteria.

III. False positive

If a message was incorrectly identified as spam, you can submit this false positive message, who will evaluate and analyze the message. Depending on the results of the analysis, the service-wide spam content filter rules may be adjusted to allow the message through.

IV. SPAM Filters

The dramatic increase of the SPAM in the past two years has created a real interest among researchers to investigate methods to combat SPAM. Researchers are presently working on the implementation of new filters that prevent SPAM from reaching their destination either by blocking it at the server level (i.e. organization with its own email server) or the client level (such as a user at home).

A SPAM filter has a set of instructions to block emails based on the nature of the content of the email.

The flexibility of a SPAM filter depends on the filtering software. SPAM filter analysis may be directed at the following:

Header analysis, here the SPAM filter will check the header of the incoming email, if the header is defined as a SPAM (for example: free gifts for you, you are the winner…etc) the SPAM filter will prevent the message from passing through to the recipient.

Address lists analysis, if the incoming email is from any unknown sender, the SPAM filter will check the email address of the sender against the address list that the recipient allows receiving messages from, and then the message will be blocked or passed through to the user.

Keyword lists analysis, the SPAM filter will check the contents of the incoming message if it has any words that could be suspicious (like: Viagra, Sex…etc), then the message will be blocked.

SPAM filters prevent SPAM from being transmitted to the recipient.

V. Black List Filter

A blacklist SPAM filter operates by creating a list of common words or phrases found in the header of the email message and domain name, which can be used to decide if an email should be prevented from passing through the SPAM filter.

VI. White List Filter

A whitelist SPAM filter is the opposite of the blacklist, and it assumes that all emails are SPAM unless they can pass through the filter. A whitelist may contain a list of email addresses that the user created to receive messages only from trusted sources. Alternatively it could be a list of domains which must be defined as legitimate before the message passes through the filter to the recipient.

VII. Bayesian Filtering

Bayesian filtering is an extension of the text classification technology. This filter is a computer program used to recognize the words in a document, and can be implemented in a SPAM filter to search the textual content of an email. Bayesian filtering method uses text categorization algorithms to determine the probability that a certain email is SPAM. The algorithms are capable of categorizing the occurrence of certain words or phrases in terms of how and where they appear in the email message, but not by their existence alone.

VIII. Bayesian Filter subject to HAM or SPAM

The Bayesian filter will be applied on <SUBJECT> field and the content <BODY> of the message. The filter scans through the message, and creates a probability of every word (spamicity). This spamicity value is assigned to each word, and ranges from 0.0 to1.0. If the spamicity value of the email message greater than or equal to 0.5 then the message containing the word is likely to be SPAM. The filter will decide if the incoming email is a SPAM email.

If the spamicity value is less than 0.5, the message containing the word is likely to be HAM.

IX. How a Bayesian Filter works?

The Bayesian filter can be used to filter the emails, the user has to create two databases with individual words and tokens (for example the #, $, *... signs, IP addresses, some specific words "Viagra" and domains…etc), gathered from two representative samples of SPAM emails (SPAM word list) and legitimate emails (HAM word list), where HAM refers to legal emails.

The filter will assign a probability value for each word or token based on how often that word or token occurs in the email message. The probability value designated to each word or token is commonly known as spamicity, and ranges from 0.0 to 1.0 as shown in Figure.

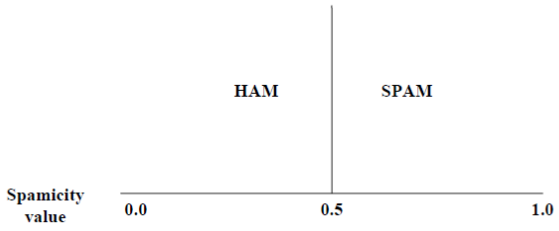

Fig. 1 Spamicity value

The spamicity could be one of the following possibilities

- Spamicity > 0.5 then the word (or the message) is most likely a SPAM.
- Spamicity < 0.5 then the word (or the message) is most likely a HAM.
- Spamicity = 0.5 then the word (or the message) is neutral, meaning that it has no effect (like "for", "what", "your"…etc).

After the SPAM and the HAM databases have been created, the probabilities of any word in the email message can be calculated and the filter will be prepared for use.

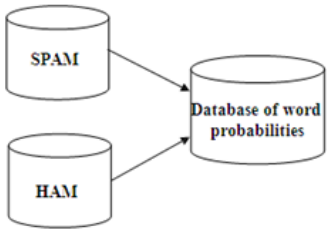

Fig. 2 Creating a word database for the Bayesian filter

When the new email message arrives, the filter will break down the incoming message into words, and calculate the spamicity value for each word, and then calculates the probability of the newly arrived message. If the probability is greater than 0.5 then the message is SPAM.

X. Design of the Proposed Filter

The architecture and the algorithm used in the proposed approach are illustrated in Figure. It can be seen that, proposed SPAM filter is composed of three filters which act in tandem. The first filter (Whitelist filter) will allow only the trusted email messages to pass through to the inbox. The second filter (Blacklist filter) will block the known SPAM messages. The third filter (Bayesian filter) will determine whether the incoming new email is going to be recognized as a legitimate or SPAM message.

The sequence of filtering methods used in proposed SPAM filter aims to:

- Speed up the process of receiving the legitimate emails, by using the whitelist method at the beginning. If a new message arrives the proposed SPAM filter will process the message against the whitelist filter, and sends the message to the inbox.

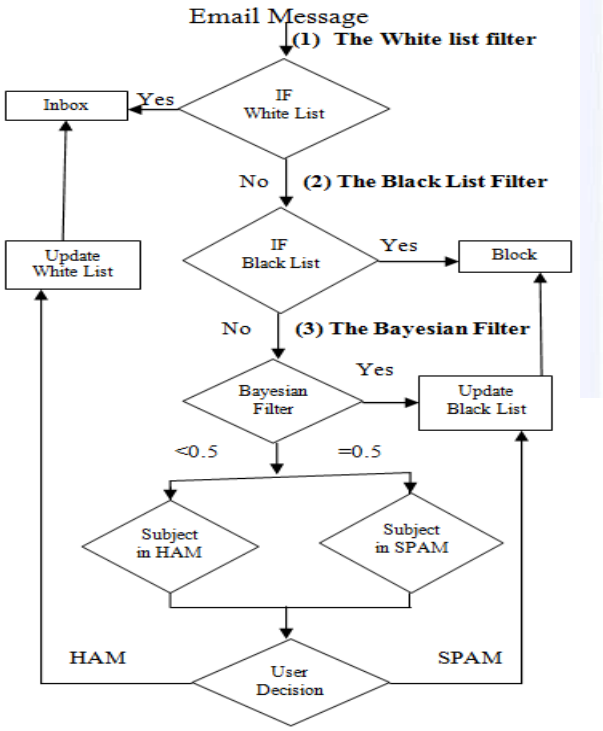

Fig. 3 The design algorithm of proposed SPAM filter

- Reduce the user intervention, and builds up an auto-update technique for both SPAM, and
- Block the known SPAM emails, by using the blacklist method. If a new SPAM message arrives the proposed SPAM filter will process the message against the second process which is the blacklist filter.

XI. Implementation of Proposed Bayesian SPAM Filter

The implementation steps of proposed SPAM filter are described below:

Step 1: Message classification

Email messages would determine one of the four program outputs below:-

- Message [A] = the message is definitely a Whitelist Message
- Message [B] = the message is not in Whitelist but subject to HAM
- Message [C] = the message is not in Whitelist but subject to SPAM
- Message [D] = the message is definitely a Blacklist Message

Step2: Variables declaration:

- String variables: to store the names of the files "blacklist.txt", "whitlist.txt", defaultStopWords.txt", "ham.txt"
- Integer variables: as counters blackListCount, whiteListCount, stopWordListCount, hamWordListCount
- Arrays: to store the content of the files blackListArray, whiteListArray, stopWordListArray, hamWordListArray.
- Boolean variable: (flag) which is to identify the status of the message.

Step3: Checking status:

- Check the number and status of the email message (flag) i.e. new or old message.
- Output the number of new email messages to the user.

Step4: Filtering processes:

- Whitelist check: If the incoming email address is found in whitelist array
  Then the message is a legitimate message. display (" [A] Definitely Whitelist "). display the email address, and the content of the message.
- Blacklist check:
  If the incoming email address found in blacklist array.
  Then the message is an illegitimate message display (" [D] Definitely a Blacklist Message ").
- Bayesian check:
  If the incoming email address is not in the whitelist or the blacklist
  Then calculate the spamicity of the entire email message.
  If spamicity < 0.5 then message is subject to HAM
  display ([B] not in the whitelist but subject to HAM)
  display (Do you want to add to whitelist Y/N)
  If (Y) then add the incoming email address to the whitelist.
  If (N) then add the incoming email address to the blacklist
  if spamicity >= 0.5 the message is subject to SPAM
  display ([C] not in the blacklist but subject to SPAM)
  display (Do you want to add to blacklist Y/N)
  If (Y) then add the incoming email address to the blacklist.
  If (N) then add the incoming email address to the whitelist.

XII. Methodology of Proposed SPAM Filter

We start with one corpus of spam and one of nonspam mail. At the moment each one has about 4000 messages in it. We scan the entire text, including headers and embedded html and javascript, of each message in each corpus. We currently consider alphanumeric characters, dashes, apostrophes, and dollar signs to be part of tokens, and everything else to be a token separator. (There is probably room for improvement here.) We ignore tokens that are all digits, and we also ignore html comments, not even considering them as token separators. We count the number of times each token (ignoring case, currently) occurs in each corpus. At this stage we end up with two large hash tables, one for each corpus, mapping tokens to number of occurrences.

Next We create a third hash table, this time mapping each token to the probability that an email containing it is a spam, which we calculate as follows :

```
(let ((g (* 2 (or (gethash word good) 0)))
      (b (or (gethash word bad) 0)))
(unless (< (+ g b) 5)
(max .01 (min .99 (float (/ (min 1 (/ b nbad))
(+ (min 1 (/ g ngood)) (min 1 (/ b nbad)))))))))
```

where word are the tokens whose probability we're calculated, good and bad are the hash tables which we created in the first step, and ngood and nbad are the number of nonspam and spam messages respectively.

We want to bias the probabilities slightly to avoid false positives, and by trial and error We've found that a good way to do it is to double all the numbers in good. This helps to distinguish between words that occasionally do occur in legitimate email and words that almost never do. We only consider words that occur more than five times in total (actually, because of the doubling, occurring three times in nonspam mail would be enough). And then there is the question of what probability to assign to words that occur in one corpus but not the other. Again by trial and error we chose .01 and .99. There may be room for tuning here, but as the corpus grows such tuning will happen automatically anyway.

The especially observant will notice that while We consider each corpus to be a single long stream of text for purposes of counting occurrences, We use the number of emails in each, rather than their combined length, as the divisor in calculating spam probabilities. This adds another slight bias to protect

against false positives. When new mail arrives, it is scanned into tokens, and the most interesting fifteen tokens, where interesting is measured by how far their spam probability is from a neutral .5, are used to calculate the probability that the mail is spam. If probs is a list of the fifteen individual probabilities, you calculate the combined

Probability thus:

```
(let ((prod (apply #'* probs)))
(/ prod (+ prod (apply #'* (mapcar #'(lambda (x)
(- 1 x)) probs)))))
```

One question that arises in practice is what probability to assign to a word you've never seen, i.e. one that doesn't occur in the hash table of word probabilities. We have found, again by trial and error, that. 4 is a good number to use. If you've never seen a word before, it is probably fairly innocent; spam words tend to be all too familiar.
We treat mail as spam if the algorithm above gives it a probability of more than .9 of being spam. But in practice it would not matter much where we put this threshold, because few probabilities end up in the middle of the range.

XIII. Conclusion

SPAM is not an easy problem to solve. SPAM has become very popular due to a variety of reasons. Unscrupulous companies and individuals can reap high rewards from unsuspecting victims without having a high ingoing and ongoing investment cost. The most commonly used method for stopping SPAM in use today is the deployment of SPAM filters the proposed SPAM filter more effective and accurate in detecting SPAM messages. This paper demonstrated that proposed SPAM filter outperformed in terms of detecting SPAM in HTML messages also, SPAM filter significantly outperformed *SpamEater©* in dealing with HTML messages.

REFERENCES

[1] Berry, D. A. (1996). Statistics - A Bayesian perspective, Duxbury Press. Bradley, P. and T. Louis (1996). Bayes and Empirical Bayes Methods for Data Analysis. London, Chapman & Hall.

[2] CAUCE (2003). "The Coalition against Unsolicited Commercial Email". Retrieved 23 April 2004, from http://www.cauce.org.

[3] CNN (2000). "AOL says hackers may have stolen credit card numbers". Retrieved 16 January 2004, from http://archives.cnn.com/2000/TECH/computing/06/17/aol.hacker.01/index.html.

[4] Comcast (2002). "How does a spammer get my e-mail address? ". Retrieved 30 October 2003, from http://www.comcast.net/help/faq/index.jsp?faq=SecuritySpam17862.

[5] Connolly, C. (2003). "NOIE proposes anti-spam legislation". Retrieved 5 December 2004, from http://consult.galexia.com/public/research/articles/research_articles- art27.html.

[6] ContentWatch (2005). "ContentWatch™ Products". Retrieved 21 November 2005, from http://www.contentwatch.com/products/emailprotect.php.

[7] COTSE (2004). "Email Filters: Overview". Retrieved 20 February 2005, from http://www.cotse.net/emailfilters.html.

[8] Crawford, E., J. Kay, et al. (2001). Automatic Induction of Rules for e-mail Classification. Sixth Australian Document Computing Symposium, Coffs Harbour, Australia.

[9] CRMToday (2002). "ContentWatch Releases a New Technology for Eliminating Unwanted E-mail". Retrieved 2 May 2004, from http://www.crm2day.com/news/ crm/EpupZkAlpZnHxuLoir.php.

[10] DCITA (2005). "Spam Act Review". Retrieved 11 April 2005, from http://www.dcita.gov.au/ie/spam_home/spam_act_revie.

[11] Christina V, Karpagavalli S, Suganya G[2010] "A study on Email SPAM Filtering Techniques" IJCA vol 12,2010.