Implementing spam detection using Bayesian and Porter Stemmer keyword stripping approaches

Conference Paper · February 2009

DOI: 10.1109/TENCON.2009.5396056 · Source: IEEE Xplore

Biju Issac
Northumbria University
101 PUBLICATIONS **660** CITATIONS

Implementing Spam Detection using Bayesian and Porter Stemmer Keyword Stripping Approaches

Biju Issac and Wendy J. Jap
School of Computing and Design
Swinburne University of Technology (Sarawak Campus)
Kuching, Malaysia
Email: bissac@swinburne.edu.my and my.keyblade@gmail.com

***Abstract*—Unsolicited or spam emails are on the rise, where one's email storage inbox is bombarded with emails that make no sense at all. This creates excess usage of traffic bandwidth and results in unnecessary wastage of network resources. We wanted to test the Bayesian spam detection scheme with context matching that we had developed by implementing the keyword stripping using the Porter Stemmer algorithm. This could make the keyword search more efficient, as the root or stem word is only considered. Experimental results on two public spam corpuses are also discussed at the end.**

Keywords-spam email; spam detection; bayesian approach; keyword stemming;

I. Introduction

The spammers are getting smarter and the challenge of developing accurate spam filters is a big challenge. E-mail spam has steadily, even exponentially grown since the early 1990s to several billion messages in a single day. Spam has frustrated, confused, and annoyed e-mail users, by wasting time and valuable resources. Laws against spam have been sporadically implemented, with some being opt-out and others requiring opt in e-mail. The total volume of spam (over 100 billion emails per day) has reached a threshold level in recent years, and is no longer growing much. The amount received by most e-mail users has decreased, mostly because of better filtering. Botnets, networks of virus-infected computers, are used to send about 80% of spam. Since the cost of the spam is borne mostly by the recipient, it is effectively postage due advertising.

E-mail addresses are collected from chat-rooms, websites, newsgroups, and viruses which harvest users' address books, and are sold to other spammers. Much of spam is sent to invalid e-mail addresses. Internet Service Providers have attempted to recover the cost of spam through lawsuits against spammers, although they have been mostly unsuccessful in collecting damages despite winning in court. Roughly, spam averages 95% of all e-mail sent [1]-[2].

The paper does the implementation of spam detection using Bayesian approach and porter stemming key word stripping approach and is tested on two public spam corpuses. The use of stem keywords makes the key word search more efficient, as the search database would be smaller.

This paper is organized as follows. Section 2 is the related works, section 3 is on porter stemmer algorithm, section 4 is on methodology used, section 5 is the experimental results and section 6 is the conclusion.

II. Related Works

Duan et al. has authored a paper in which they analyze a two-month trace of more than 25 million emails received at a large US university campus network, of which more than 18 million are spam messages. They characterize the spammer behavior at both the mail server and the network levels. They also correlate the arrivals of spam with the BGP route updates to study the network reachability properties of spammers. Their significant findings are: (a) the majority of spammers (93% of spam only mail servers and 58% of spam only networks) send only a small number of spam messages (no more than 10); (b) the vast majority of both spam messages (91.7%) and spam only mail servers (91%) are from mixed networks that send both spam and non-spam messages; (c) the majority of both spam messages (68%) and spam mail servers (74%) are from a few regions of the IP address space (top 20 "/8" address spaces); (d) a large portion of spammers (81% of spam only mail servers and 27% of spam only networks) send spam only within a short period of time (no longer than one day out of the two months); and (e) network prefixes for a non-negligible portion of spam only networks (6%) are only visible for a short period of time (within 7 days), coinciding with the spam arrivals from these networks [3].

Chun-Chao et al. finds that due to fast changing of spam techniques, they argue that multiple spam detection strategies should be developed to effectively against spam. Among many others, (naive) Bayesian filter is one of those being proposed. While there are some early works being done on Bayesian approaches for spam detection, it is unknown if such a simple approach is still effective to filter spam mail today. They re-evaluate the Bayesian approach with a public spam database. and found that Bayesian approach can achieve high spam detection rate for plain-text mail [4].

Tan et al. in their paper was inspired by human immune system, a concentration based feature construction (CFC) approach which utilizes a two-element concentration vector as the feature vector is proposed for spam detection in this paper. In the CFC approach, concentrations are constructed by using dasiaselfpsila and dasianon-selfpsila gene libraries, respectively, and subsequently are used to form a vector with two elements of concentrations for characterizing the e-mail efficiently. The classification of the e-mail is considered as an optimization problem aiming at minimizing a formulated cost function. A clonal particle swarm optimization (CPSO) algorithm proposed by the leading author is also employed for this purpose [5].

Matsumoto et al. in their paper describe the results of an empirical study on two spam detection methods: support vector machines (SVMs) and naive Bayes classifier (NBC). To conduct the study, they implement the NBC and choose to use the SVMlight, an application of SVMs developed by Thorsten Joachims. The NBC and the linear SVMs with different C parameters are trained on a set of 2000 emails with 1000 spams and 1000 nonspams, and are tested on 200 new emails with 100 in each class. A program is written that converts emails into feature vectors using both TF and TF-IDF term weighting methods. The evaluation criteria include accuracy rate, recall, precision, miss rate, and false alarm rate [6].

Begriche et al. in their paper deals with Bayesian models applied to anti-spam. In most anti-spam related researches, authors assume that the probability of spam message is equal to 0.5, which is unrealistic. This pushes us to define a prior and a posterior probability laws to enhance the spam detection and increase the reliability decision [7].

For email classification as spam or non-spam, naive bayes classification was used in several systems [8]-[9].

III. PORTER'S STEMMER ALGORITHM

Information Retrieval (IR) is primarily a matter of selecting which documents or specific text(s) in a collection should be retrieved to satisfy a user's request for information. The user's information need is expressed through a query. The retrieval decision is made by comparing the terms of the query with the index terms (important words or phrases) appearing in the document itself. In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. For this reason, a number of so-called stemming algorithms, or stemmers, have been developed, which attempt to reduce a word to its stem or root form. Thus, the key terms of a query or document are represented by stems rather than by the original words.

Word stemming is an important feature supported by present day indexing and search systems. The idea is to improve recall by automatic handling of word endings by reducing the words to their word roots, at the time of indexing and searching. Stemming broadens our results to include both word roots and word derivations. It is commonly accepted that removal of word-endings (sometimes called suffix stripping) is a good idea; removal of prefixes can be useful in some subject domains.

The Porter Stemmer is a conflation Stemmer developed by Martin Porter at the University of Cambridge in 1980. Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. It is widely used. The Stemmer is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. This Stemmer is a linear step stemmer and it has five steps applying rules within each step. Within each step, if a suffix rule matched to a word, then the conditions attached to that rule are tested on what would be the resulting stem, if that suffix was removed, in the way defined by the rule. Porter's Algorithm works based on number of vowel characters, which are followed be a consonant character in the stem (measure), must be greater than one for the rule to be applied [15].

Once a rule passes its conditions and is accepted the rule is implemented and the suffix is removed and control moves to the next step. If the rule is not accepted then the next rule in the step is tested, until either a rule from that step is executed and control passes to the next step or there are no more rules in that step whence control moves to the next step. This process continues for all five steps, the resultant stem being returned by the Stemmer after control has been passed from step five. See figure 1 to see the stemming process and table 1 for sample suffix mapping. [10]

Like mentioned before, the rules in the Porter algorithm are separated into five distinct steps numbered from 1 to 5. They are applied to the words in the text starting from step 1 and moving on to step 5.

The step 1 deals with plurals and past participles. The subsequent steps are much more straightforward. E.g. checked → check, laughing → laugh etc. The step 2 deals with pattern matching on some common suffixes. E.g. happy → happi, relational → relate, callousness → callous etc. The step 3 deals with special word endings. E.g. triplicate→ triplic, hopeful→ hope etc. The step 4 checks the stripped word against more suffixes in case the word is compounded. E.g. revival → reviv, allowance→ allow, inference→ infer etc. The step 5 checks if the stripped word ends in a vowel and fixes it appropriately. E.g. probate → probat, cease → ceas, controll → control etc. The algorithm is careful not to remove a suffix when the stem is too short, the length of the stem being given by its measure (say, m). There is no linguistic basis for this approach.

We want to implement stemming to reduce the keyword search and thus to improve the efficiency of spam detection.

The technique that we adopted to implement the spam filter is based on the content-filtering method, in which we analyze the content of inbound emails to find out particular characteristics that differentiate between legitimate and illegitimate emails. Several implementations of content-filtering method have been proposed by others, such as keyword analysis, Bayesian analysis, and heuristic approaches. In our research, we propose to use the Bayesian approach, because the previous work has told that the quality of using Bayesian approach is quite promising as in [11].

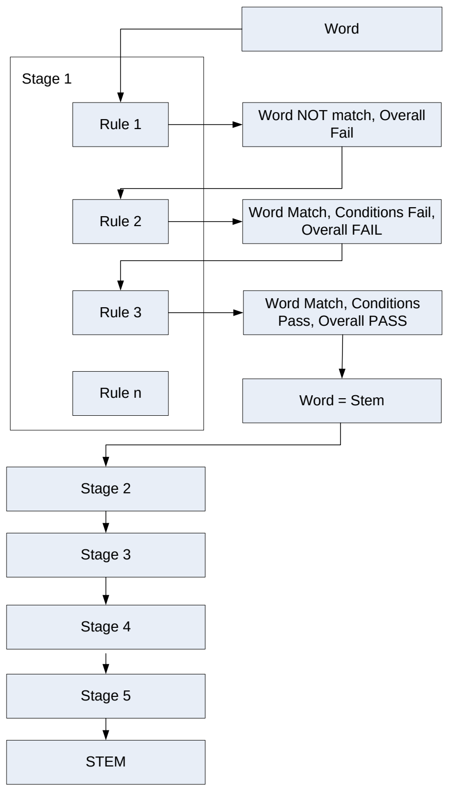
Word
Stage 1
Rule 1
Word NOT match, Overall Fail
Rule 2
Word Match, Conditions Fail, Overall FAIL
Rule 3
Word Match, Conditions Pass, Overall PASS
Rule n
Word = Stem
Stage 2
Stage 3
Stage 4
Stage 5
STEM

Fig.1. The stemming process [16]

TABLE I
SUFFIX MAPPING EXAMPLE

Original suffix	Changed suffix
izer	ize
iser	ize
ization	ize
isation	ize
ation	ate
ator	ate
alism	al
iveness	ive
fulness	ful
ousness	ous

IV. METHODOLOGY USED

We present three kinds of Bayesian-based approaches, in which we set some additional features in every one of them to differentiate between one and another. Each of the method will be described below and has been implemented in Java.

A. *Simple Bayesian Approach*

Basically, the Bayesian approach identifies spam from other legitimate emails by considering the number of occurrences of particular characteristics that resemble a spam or a ham (non-spam email). Therefore, we are looking at the use of certain keywords in every mail. If a keyword in an email is frequently found in spam, it increases the probability that the mentioned email is also a spam. On the other hand, when a keyword is found in most non-spam mail, it decreases the probability that the mentioned email is a spam.

This technique requires training with known spam and non-spam emails. During the training phase, the spam filter keeps record of the words which are often used in both spam and non-spam emails. The number of occurrences of these words will eventually be used to determine the probability if a mail is spam or non-spam.

When the spam filter accepts a test mail, it extracts the keywords and gives probability score to every keyword based on the number of occurrences that have been gathered during training stage. The Bayesian probability *p(k)* for keyword *k* is given as in the following equation:

$$p(k) = \frac{s(k)}{s(k) + ns(k)} \qquad -(1)$$

where *s(k)* is the number of spam email with keyword *k* and *ns(k)* is the number of non-spam emails with keyword *k*.

We realized that there are several words that have to be ignored, such as *I, you, can, except, here,* etc. Thus, these words are excluded in the scoring. This approach doesn't only take account of single keyword, but also multiple keywords during (e.g. a phrase). However, the simple Bayesian approach gives equal weight to both of them using the same equation (1).

Moreover, we only extract one hundred words from each mail to be taken into the calculation. According to [1], if we are using all the keywords, we may miss identifying spam with long content. In practice, these hundred words will be trimmed down into 15 to 30 keywords only, because some of them are ignored. We also consider that keywords, which are extracted from the subject header of an email, should be given more weight, because the content of most emails can be identified by only looking at their subject title.

The probability score of each keyword will be summed up. The total score will then be compared to a threshold value. When the total score exceeds the threshold value, we decide that the corresponding mail is a spam, and vice versa.

B. Improved Bayesian with Multiple Keywords

The only difference of this approach from the previous one is that multiple keywords are given more weights than single keyword. Therefore, the presence of multiple keywords increases the probability that a mail is a spam or non-spam. However, multiple keywords that consist of two words, three words, or more, are still treated equally the same. We introduce the *MK_WEIGHT* constant which is used to multiply the score of a multiple keyword produced by equation (1). Hence, the total score for a test mail is the sum of the single keyword's score and the multiple keyword's score.

C. Improved Bayesian with Keyword Context Matching

Based on the previous method, we add a keyword context score to the total score. We perceive that a word may have many contexts. By finding out the context of each word in a mail, we will further improve the accuracy of identifying spam.

Context is a set of remaining keywords that is mapped to every keyword chosen. For example, if we have collected *n* keywords from a test mail (i.e. *keyword-1, keyword-2, keyword-3, ..., keyword-n*), *keyword-1* has a context of [*keyword-2, keyword-3, ..., keyword-n*], while *keyword-2* has a context of [*keyword-1, keyword-3, ... keyword-n*]. For every keyword context, we count how many words in the context that match with the existing known spam keywords (retrieved during the training). Therefore, we calculate context matching percentage (CMP) as:

$$CMP = \frac{\text{no. of matching keywords}}{\text{no. of keywords in the context}} \qquad -(2)$$

Therefore, the total spam score for an unidentified email is:

Total spam score = single-keyword spam score + multiple keyword spam score + CMP $-(3)$

During the training stage, the spam filter creates keyword bank which stores all keywords extracted from known spam and non-spam emails. For comparison purpose, we implemented the Porter's stemmer algorithm in the word extraction phase. This algorithm returns the stem of a word by stripping the suffixes embedded on the word. For example, insertion becomes insert, books becomes book, etc. [12]-[13].

V. EXPERIMENTAL RESULTS ON SPAM CORPUSES

In order to test our spam filter, we used two public spam corpuses which are available on the internet – Ling Spam Corpus and Enron Spam Corpus. Ling-spam corpus consists of 2,413 non-spam emails and 481 spam emails. Enron-spam corpus consists of 16,545 non-spam emails and 17,171 spam emails [14].

In the Ling-spam corpus used (under bare directory), it contained contains 10 subdirectories (part1, ... part10). These correspond to the 10 partitions of the corpus that were used in the experiment. The 9 parts (part1 to part 9) were used for training and one part was used for testing (part 10). Later, all possible combinations of folders were used – nine for training and one for testing. Each one of the 10 subdirectories contains spam and legitimate messages, one message in each file.

For Ling Spam Corpus, the experiments were done using the parameter values as follows.

- Spam threshold for simple Bayesian approach = 0.15
- Spam threshold for improved Bayesian approach = 0.15
- Spam threshold for improved Bayesian + CMP = 0.2
- MK_WEIGHT constant = 50
- Weight constant for keywords at the subject header = 50

In Enron corpus, it was organized into 6 folders. Each time five folders are used for training and the remaining one was used for testing. In our implementation, we extracted only the first 100 keywords from all the mails for spam score analysis [15]. Table 2 and 3 show the results of experiments with Ling-spam corpus and Enron Corpus.

For Enron-spam corpus, the experiments were done using the parameter values as follows.

- Spam threshold for simple Bayesian approach = 0.57
- Spam threshold for improved Bayesian approach = 0.59
- Spam threshold for improved Bayesian + CMP = 0.7
- MK_WEIGHT constant = 50
- Weight constant for keywords at the subject header = 50

TABLE II. LING-SPAM CORPUS EXPERIMENT RESULT WITH PORTER STEMMER ALGORITHM

Bayesian with single keyword		Bayesian with multiple keywords		Bayesian with multiple keywords and context matching		Train and Test folders
False +ve %	False -ve %	False +ve %	False -ve %	False +ve %	False -ve %	
13.22	2.08	9.50	4.17	6.61	4.17	2-10 and 1
12.03	4.17	5.81	6.25	6.22	2.08	1, 3-10 and 2
7.88	2.08	6.22	0.00	4.98	0.00	1-2, 4-10 and 3
11.62	2.08	7.88	0.00	7.05	0.00	1-3, 5-10 and 4
2.89	2.08	1.65	2.08	0.83	2.08	1-4, 6-10 and 5
2.49	2.08	0.83	2.08	1.66	2.08	1-5, 7-10 and 6
6.22	0.00	4.15	2.08	4.98	4.17	1-6, 8-10 and 7
15.77	0.00	10.37	0.00	11.20	0.00	1-7, 9-10 and 8
6.22	6.25	3.73	4.17	3.32	4.17	1-8, 10 and 9
5.79	10.20	2.89	12.24	2.89	14.29	1-9 and 10
8.41 (avg)	**3.10 (avg)**	**5.31 (avg)**	**3.31 (avg)**	**4.97 (avg)**	**3.30 (avg)**	

As mentioned before, the Porter stemming algorithm (or ‘Porter stemmer’) is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems. We used this algorithm to do spam detection using stem keywords, rather than using the full keywords, on the above two corpuses and it yielded the results as in table 2 and 3. The number of keywords stored would also be less in number, if this option is not used. The accuracy graphs for Ling Spam and Enron corpuses are as shown in figure 2 and 3 respectively.

TABLE III. ENRON-SPAM CORPUS EXPERIMENT RESULT WITH PORTER STEMMER ALGORITHM

Bayesian with single keyword		Bayesian with multiple keywords		Bayesian with multiple keywords and context matching		Train and Test folders
False +ve %	False -ve %	False +ve %	False -ve %	False +ve %	False -ve %	
10.57	19.33	8.47	14.47	4.55	18.13	2-6 and 1
14.77	14.37	12.27	4.01	11.35	2.27	1, 3-6 and 2
8.97	12.53	6.11	6.20	3.56	7.00	1-2, 4-6 and 3
2.00	32.34	1.93	24.29	0.73	28.34	1-3, 5-6 and 4
3.07	33.88	2.60	17.36	2.07	17.47	1-4, 6 and 5
4.27	31.96	2.87	18.42	1.80	20.84	1-5 and 6
7.27 (avg)	**24.07 (avg)**	**5.71 (avg)**	**14.13 (avg)**	**4.01 (avg)**	**15.68 (avg)**	

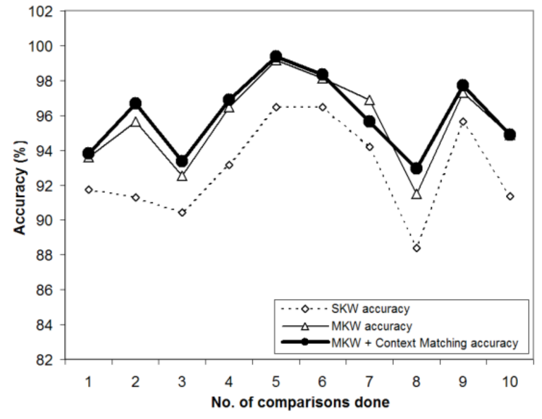

Fig.2 The graph for LING-SPAM corpus showing the spam detection accuracy for the three approaches (single keyword, multiple keywords, and multiple keywords with context matching) along with keyword stemming.

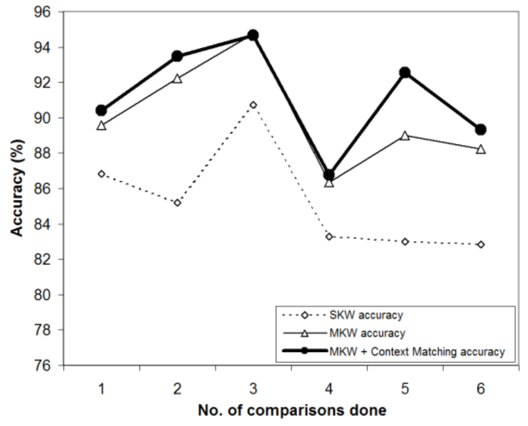

Fig.3 The graph for ENRON corpus showing the spam detection accuracy for the three approaches (single keyword, multiple keywords, and multiple keywords with context matching) along with keyword stemming.

VI. Conclusion

The use of Porter's stemmer algorithm to strip every keyword to its stem does improve the filter's efficiency in terms of reducing the keyword searches and also generally improves the accuracy marginally. We can also see that giving more weight for extracted multiple keywords helps to boost the accuracy of the filter as being implemented by the improved Bayesian approach. However, the use of context matching percentage in the third approach doesn't produce significant improvement, but it can still improve with different multiple keyword weights, fine-tuned stemming etc. That would be our future work on this research.

References

[1] S. Kiritchenko and S. Matwin, "Email classification with co-training," in the Centre for Advanced Studies on Collaborative Research, Toronto, Ontario, Canada, 2001, pp.1-8.

[2] K. J. Chan and J. Poon, "Co-training with a single natural feature set applied to email classification," Proc. of IEEE International Conference on Web Intelligence, 2004.

[3] Z. Duan, K. Gopalan, X. Yuan, Behavioral Characteristics of Spammers and Their Network Reachability Properties, IEEE International Conference on Communications, 2007, pp.164-171.

[4] Y. Chun-Chao; C. Soun-Jan, Revisit Bayesian Approaches for Spam Detection, The 9th International Conference for Young Computer Scientists, 2008, pp.659-664.

[5] Y. Tan, C. Deng and G. Ruan, Concentration based feature construction approach for spam detection, International Joint Conference on Neural Networks, 2009, pp.3088-3093.

[6] R. Matsumoto, D. Zhang and M. Lu, Some empirical results on two spam detection methods, Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, 2004, pp.198-203.

[7] Y. Begriche, and H. Labiod, A Prior Distribution for Anti-spam Statistical Bayesian Model, International Conference on Network and Service Security, 2009, pp.1-5.

[8] K. Schneider, "A comparison of event models for naive bayes anti-spam e-mail filtering", Proc. of 11th Conference of the European Chapter of the Association for Computational Linguistics, 2003.

[9] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos, "Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach," Proc. of 4th PKDD's Workshop on Machine Learning and Textual Information Access, 2000.

[10] M. F. Porter, An algorithm for suffix stripping, Program, 14(3), pp.130-137, 1980.

[11] What is Porter Stemming?, Computing Department, Lancaster University. Available: http://www.comp.lancs.ac.uk/computing/research/stemming/general/porter.htm, Accessed April 2009.

[12] P. Graham, "Better Bayesian Filtering", 2003. Retrieved 2 May 2005, [Online]: http://www.paulgraham.com/ better.html

[13] B. Issac and V. Raman, "Implementation of Spam Detection on Regular and Image based Emails - A Case Study using Spam Corpus", Proc. of MMU International Symposium on Information and Communication Technologies, 2006, pp.431-436.

[14] Internet Content Filtering Group, Spam Corpora, [Online]: http://www.iit.demokritos.gr/skel/i-config/

[15] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras and C.D. Spyropoulos, "An Evaluation of Naive Bayesian Anti-Spam Filtering". Proc. of Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, Spain, pp. 9-17, 2000.