An Evaluation and Analysis of Static and Adaptive Bayesian Spam Filters

Qussai Yaseen[1], Yaser Jararweh[2], Khanh Nguyen Viet[3]

[1] Department of Computer Information Systems, Jordan University of Science and Technology, Jordan
[2] Department of Computer Science, Jordan University of Science and Technology, Jordan
[3] Epic Corporation, Madison, WI, USA
{qmyaseen, yijararweh}@just.edu.jo, knguyenv@uark.edu

Abstract

Spams and spamming methods are increasing vastly and getting complicated due to the rapid growth in networks, communications and technologies. Therefore, spam filters need to be tested continuously to evaluate their capabilities and efficiency in detecting and preventing spams. This paper discusses spams filtering problem using Bayesian classifier. It shows how using a combination of black and white lists and a customized spam filter based on users' feedback can enhance the performance of Bayesian classifier. The paper evaluates three models of spam filters which are Static Bayesian Spam Filter, Light Adaptive Bayesian Spam Filter, and Enhanced Adaptive Bayesian Spam Filter. The experiments demonstrate that Enhanced Adaptive Bayesian Spam Filter, which is the one that uses black/white lists and users' feedback, has the highest performance.

Keywords: Spam filter, Bayesian classifier, Security, Classification, White lists, Black lists

1 Introduction

Spams are still among the most annoying things in Internet. As the world has become more connected due the rapid growth in networks and communications, the number of spams and spammers has increased, and the spamming techniques have gotten complicated. The need to effective spam filters able to stop spams attacks is increasing. Similarly, existing spam filters need to be improved to cope with new types of spam attacks. Therefore, existing spam filters need to be tested continuously to evaluate their capabilities and efficiency against spams.

Spam filters are classified into several categories such as origin-based filters, content-based filters and collaborative filters [1-3]. Origin-based filters such as [4-5] use network and routing information of a message to check it is spam or not. The aforementioned type may use different methods to detect spams such as using black and white lists. A black list contains a set of untrusted IP addresses. Any message that has an IP address that exists in the black list is considered a spam. However, this approach suffers from a number of flaws such as the difficulty of keeping a black list up-to-date, and the possibilities of using "Zombie" computers to avoid this type of filters. Another model that belongs to this category is using whitelists to store trusted addresses. Any message with an IP address that exists in the whitelist is considered a legitimate address. However, the size of such kind of lists grows very rapidly which makes them difficult to be maintained and used.

Content-based filters [6-8] use the content of a message to identify whether if it is spam or not. This type may use messages headers or common keywords in messages. Header analysis approach focuses on the information contained in the header of a message, including routing and subject information. Collaborative filters [9-10] combine the experiences of many users to help identify spams. Spam is often sent to bulk, therefore, when a user classifies a message as a spam, other users can rely on that classification without analyzing the message themselves.

This paper evaluates three Bayesian Spam Filters, which are Static Bayesian Spam Filter, Light Adaptive Bayesian Spam Filter, and Enhanced Adaptive Bayesian Spam Filter. The last two models are based on the Graham Approach who used a combination of black-lists, white-lists and Bayesian Classifier to detect and prevent spams. The paper demonstrates the model, the algorithm and the simulations that show and compare the efficiency of the three models.

The rest of the paper is organized as follows. Next section discusses some related work. Section 3 introduces the State of Art. Section 4 discusses the Adaptive Bayesian Spam Filter. Section 5 demonstrates the experiments and analysis. Section 6 concludes the work and introduces the future work.

*Corresponding Author: Qussai Yaseen; E-mail: qmyaseen@just.edu.jo
DOI: 10.3966/160792642018081904005

2 Background

Spam filtering is similar to text classification; it is a supervised learning approach that depends on a predefined set of labeled messages (spam or not spam) to predict the class of unlabeled messages [11]. An extensive research has been performed in designing spam filters or in evaluating and comparing spam filters. The goal is to reach an optimal design that effectively detects and prevents spam messages with minimum number of false negatives and false positives. Bayesian spam filter is one of the most effective spam filters. In this section, we briefly introduce some work in improving Bayesian spam filters or in comparing it to other spam filters.

Sahami et al. [7] proposed a machine learning algorithm for spam filtering. They used a Naïve Bayesian Classifier that was trained on a manually categorized spams and hams (not spams). They showed that Bayesian filter has an impressive performance on unseen incoming messages. The filter needs a training set of spam and legitimate messages (hams). It extracts three sets of words (tokens), which are words that occur only in spam messages, words that exist only in legitimate messages, and words that occur in both messages (spams and hams). Based on these sets, incoming unknown messages are classified to either spams or hams. Unlike Sahami, Androutsopoulos et al. [12] used ten-fold cross-validation to reduce the prone to random validation. Moreover, they discussed the effect of attribute-set size, training-corpus size and stop-lists on the filtering process. Vikas et al. [13] claimed that content-based filters that use Bayesian approach alone is not sufficient to function as a spam filter due to the large number of false positives. Therefore, to obtain an optimal spam filter, they suggested the use of a lemmatizer in Bayesian spam filter, where a lemmatizer is a stop-list and integration with other techniques such as the blacklist and rule-based methods.

Many researchers conducted experiments to compare and evaluate spam filters such as Cormak and Bratko [14] who showed that SVM and Naive Bayesian outperforms Knn on full email messages. Niimi et al. [15] evaluated both Bayesian and SVM spam filters and showed that these filters have high performance in detecting spams. Moreover, they showed that embedding URL pre-fetch method improves the performance of Bayesian filter. Androutsopoulos et al. [16] compared Naive Bayesian and Keyword-Based Spam Filters, and showed that the Naive Bayesian filter greatly outperforms the keyword-based filter, even with very small training set.

Some researchers evaluated spam filters on short messages such as Zelikovitz and Hirsh [17] and Healy et al. [18]. The problem of short messages is that they consist of few words composed of abbreviations and idioms. The authors in [18] compared Knn and SVM, and Naïve Bayesian Classifiers on SMS messages and hotel comment forms. Based on their results, SVM and Naive Bayesian considerably outperformed Knn. Similarly, Mahmoud and Mahfouz [19] proposed an approach for filtering SMS messages. The authors used a model based on Artificial Immune System (AIS) that uses some features such as Phone Numbers, Spam Words and Detectors to detect spams.

An interesting work was performed by Metsis et al. [20] who compared five different versions of the Naive Bayes classifier. They claimed that the best performance was achieved by two Naïve Bayesian NB versions that have been used less in spam filtering, which are Flexible Bayes and the multinomial Naïve Bayesian with Boolean attributes.

Bayesian filters have shown robust and high precision approach in filtering data. In addition to Bayesian spam filter, which is the scope of this paper, Bayesian filters was used in different research areas such as particle filtering [21] and adaptive systems.

3 State of Art

A Bayesian spam classifier is similar to documents classification using Bayesian theorem. The model discussed in this paper is called an adaptive classifier since the Bayesian spam filter for a user is updated continuously depending on the user's feedback about the emails he/she receives. Basically, Bayesian classifier is used to filter emails using the following formula:

$$P(S \mid W) = \frac{P(W \mid S) * P(S)}{(P(W \mid S) * P(S) + P(W \mid H) * P(H))} \quad \textbf{(1)}$$

Where P(S|W) is the probability that an email is a spam given that it contains the word W, P(W|S) the probability that a word W appears in spam emails, P(W|H) is the probability that a word W appears in ham (non-spam) emails, P(S) is the probability that an email is a spam message without any given information and P(H) is the probability that an email is a ham message without any given information.

P(W|S) and P(W|H) are computed based on the training set used in the Bayesian Classifier, where P(W|S) is computed as the ratio of the number of spam messages that contain the word W to the total number of spam messages, and P(W|H) is the ratio of the number of ham messages that contains the word W to the total number of ham messages. P(S) and P(H) depends on the assumption of the probability of these values in the real word. Both of them can be considered as 50% if we assume that the probability of the spam and ham emails are the same, or they can be considered as 30% and 70% respectively depending on the assumption that the percentage of spam emails and ham emails on a person's email box is 30% and 70%.

To compute the probability that a given email D is a spam or ham, the probabilities of all words in a document is combined as follows:

$$P(S \mid D) = \frac{\prod_{i=1}^{n} P_i}{\prod_{j=1}^{n} P_j + \prod_{j=1}^{n} (1 - P_j)} \quad \textbf{(2)}$$

Where $P_i=P(S|W_i)$, and $D = \{W_1, W_2, \ldots, W_n\}$.

Different approaches for using Bayesian classifier as a spam filter have been suggested. One approach is suggested by Sahami et el. [7], where each message is represented by a vector space model in which each dimension of this space is corresponding to a given word in the entire messages corpus. Therefore, each message can be represented as a binary vector denoting which words are found or not in the message. Therefore, the classifier is used to detect junk emails after the classifier learns from a pre-classified set of training messages. Sahami's approach used different features for classifying emails such as using single words learnt from a set of pre-classified junk mails, specific phrases like "Free!" or "only $", domain-specific non-textual features and the percentage of non-alphanumeric characters in the subject of a mail message like "$$$big money$$$". Single word features are extracted in the learning phase using pre-classified emails. Learning phase is performed by removing words that appear fewer than three times, computing the mutual information between each feature and each class (spam or ham), giving each feature a rank that represents how much the feature is close to a given class, and finally, selecting the top 500 features to build the classifier. After the learning process completes, incoming emails can be classified using the features learnt from the learning phase.

Graham [8] used a different style in applying Bayesian classifier as a spam filter. His approach starts with two equal corpuses of spam messages and ham messages. After that, the spam corpus is scanned to count the frequencies of words and to store the frequencies in a hash table. Next, the ham corpus is scanned to count the frequencies of words and to store the frequencies in a hash table. Then, a third hash table is created and each token (word) is mapped to the probability that an email containing it is a spam using the following formula.

$$\Pr(S \mid W) = \frac{((B(W)/(Nbad))}{(G(W)/(Ngood) + (B(W)/Nbad)} \quad \textbf{(3)}$$

Where B(W) is the frequency of the word W in the spam hash table, *Nbad* is the number of spam messages, G(W) is the frequency of W in the ham hash table and *Ngood* is the number of ham messages.

To test an incoming email, the email is tokenized and the probability of the words is extracted. Based on the probabilities, the top 15 words are chosen. After that, the probability of a spam is computed using formula 2. If the result is greater than 0.9, the email is considered a spam. Otherwise, it is as a ham. To enhance the efficiency of the filter, Graham considered words that occur more than five times only. In addition, he chose a probability of 1% and 99% for words that occur in one corpus and not occur in the other corpus. Moreover, he assigned a probability of 40% for words that not exist in the two corpuses. These values were chosen by trial and error. Graham suggested that each user should build his/her spam corpus. Thus, the spam filter is customized by building a profile for each user. This approach provides three advantages. Firstly, it makes filters more effective. Secondly, each user defines his precise definition of spam. Finally, it is difficult to spammers to tune mails to get through filters since there are many individuals databases. The work in this paper is based on Graham's ideas and uses black and white lists in the filtering process. The model is discussed in the next section, and the evaluation of the model is demonstrated in sections 4.

4 Adaptive Bayesian Spam Filter

Each user should have his own definition about spam, and a spam for a user may not be a spam for others. However, some emails are considered spams for all users. Therefore, in order to construct a successful spam filters, content-based filtering, whitelist and blacklist should be combined, where a *whitelist* is defined formally as follows.

Definition 1 (Whitelist). A list $WL = \{ID_1, ID_2, \ldots, ID_n\}$, where ID_i is an email address, is called a whitelist for a user U if for all $ID_k \in WL$, ID_k is a trusted email address according to U.

A *whitelist* indicates that a user trusts a number of senders from whom a spam will not be sent. Thus, using whitelists increases the precision of the filter and reduces the amount of processing time needed for filtering. Whitelists can be built by keeping a list of every address to whom the user has ever sent an email. In addition, *blacklists*, which contain the addresses of senders from whom the user does not like to receive, can be used. A *content-based filtering* is used to check the contents of the email to decide whether an email is a spam or not when the address of the sender is not in the blacklist and the whitelist of the user. Some researchers such as Graham [8] talked about using whitelists, blacklists and contents-based filtering in order to prevent spams. In this paper, we introduce a model for an adaptive Bayesian spam filter that uses the aforementioned ideas, and show how we can combine them to prevent spams with the least processing time.

Algorithm 1 discusses the model in details. The algorithm uses a general (global) blacklist that blocks well known spam emails and addresses common spams for all users (step 2-4). If a sender's address of an email is not in the global black list, the email is sent to the *customized spam filter engine* (step 5). In this phase, the email is blocked if the sender's address exists in the User-Black-List (steps 6-8). Otherwise, the email is checked against the User-White-List. If the sender's address exists in the User-White-List, the email is considered a ham (not spam), and is forwarded to the user (steps 9-12). However, if the sender's address is not in the User- White-List, the email is forwarded to the Bayesian Spam Filter Engine (step 13). The structure of the aforementioned engine is discussed in section 5. If the engine considers the email a spam, the sender's address is added to the User-Black-List, and the email is rejected (steps 14-17). Otherwise, the email is forwarded to the user. After finishing this phase, the algorithm asks the user to send his/her feedback about the emails that are considered hams. If the user discovers that one of the forwarded emails is a spam, the sender's address of that email is added to the User-Black-List, and the tokens of the email are added to Spam-Tokens-Database (steps 23-25). The latter steps represent the learning phase, where the engine is updated to reject those emails that were previously considered as hams using the older version of Spam-Tokens-Database. However, if the user confirms that the email is a ham, the sender's address is added to the User-White-List (if it is not there already) (steps 26-27). These steps are used to reduce the processing time needed to filter incoming emails. Figure 1 demonstrates the model based on algorithm 1. Hence, the model shows the customized Bayesian Spam Filter Engine and does not contain the Global Black List.

Algoirthm 1. Adaptive Bayesian Spam Filter

Input: User U, UserWhitelist (UWL) ={}, UserBlacklist (UBL) = {}, GlobalBlacklist (GBL) = {S_1, S_2,….., S_n}, SpamTokensDatabase (STD), Ham = {}.

Method: Use Global Black list and User Black list to reject spams immediately, use User White list to forward hams immediately, and use Bayesian Spam Filter Engine to detect spams from the uncertain emails.

```
1.  For each E  // E is an incoming Email.
2.    If SenderAddress(E) ∈ GBL  // The sender in the Global Black List
3.        E ← Spam                  // E is a spam
4.        Reject E                  // E is filtered as a spam
5.    Else                          // Forward to the customized spam filter engine
6.        If SenderAddress(E) ∈  UBL(U) // The sender in the User Black List
7.          E ←Spam
8.          Reject E
9.        Else if SenderAddress(E) ∈  UWL(U) // The sender in the User White List
10.         E ←Ham                  // E is a ham (not spam)
11.         Ham = Ham ⋃ {E}     // Add E to ham list
12.         Forward E to U        // Forward the email E to the user U
13.       Else                    // Send E to the Bayesian Spam Filter Engine (BSFE)
14.         If Check (E, BSFE) = True    // Using Bayesian Spam Filter Engine (BSFE), E is a spam
15.           E ← Spam
16.           UBL = UBL ⋃ SenderAddress(E)    // Add the sender address to the user black list
17.           Reject E
18.         Else                    // Using Bayesian Spam Filter Engine (BSFE), E is not a spam
19.           E ← Ham
20.           Ham = Ham ⋃ {E}
21.           Forward E to U
22.  For Each E in Ham                  // Checking the feedback of the user
23.    If FeedBack(U, E) = Spam         // if the user marked it as a spam
24.      STD = STD ⋃ Tokens(E)          // Update the spam tokens database SED
25.      UBL = UBL ⋃ SenderAddress(E)   // Add the sender address to the user black list
26.    Else                             // if the user didn't mark it as a spam
27.        UWL =UWL⋃SenderAddress(E)// Add sender address to user white list if it is not there
```

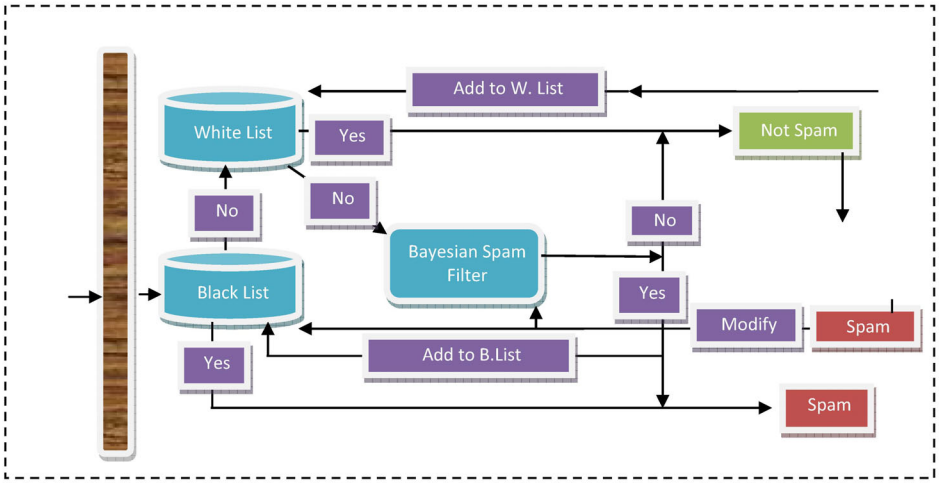


Figure 1. An adaptive Bayesian spam filter model

5 Experiments

5.1 Program Structure

Evaluating the model needs an interactive interface that enables users to send their feedback about the arrived emails. Therefore, the program was designed with an interactive web-interface. This is actually the adaptive learning feature. After letting the system check the email, the user can click on the "This is a SPAM" button to confirm that the email is actually a spam. By doing that, the entered email will also be added to the spam-corpus, and the program will "learn" one new sampling of a spam regarding to that user and become more efficient on spam-detection. Similarly, the user can click on "NOT SPAM" button to clarify that the message is a good (normal) email. The email will be added to the ham-corpus and will be useful for non-spam detection in the future.

The program uses one Black-list, one White-list, and two wordlist dictionary files. The first dictionary file (kDict.fin) stores all words (terms), along with the occurrence frequency of each word appears in the spam ham corpuses. The second dictionary file (kWord.fin) stores a pair of word – spam value which indicates that this word is more likely used in spam or ham message. The program consists of two server-webpages and four C++ programs:

(1) Token (ktoken.lex) to tokenize (separates) all the terms in the spam and ham corpus.

(2) kBW (kBWList.lex) to write the senders' addresses to black or white lists.

(3) kCalc (kCalculate.cpp to count how many messages in the Spam and Ham corpus.

(4) kCheck (kCheck.lex) to calculate the overall spam-probability of the email.

The filtering process consists of two phases: the *tokenizing (preparation) phase* and the *retrieval (checking) phase*, which are discussed in the following subsections.

The tokenizing phase uses the *Token* and *kBW* programs to generate the two dictionary files and the Black/White lists. As shown in Figure 2, the Spam and Ham messages are used by two programs, which are Token and kBW, to produce the dictionary files and the Black / White lists. The Token program has two sub-processes as follows:

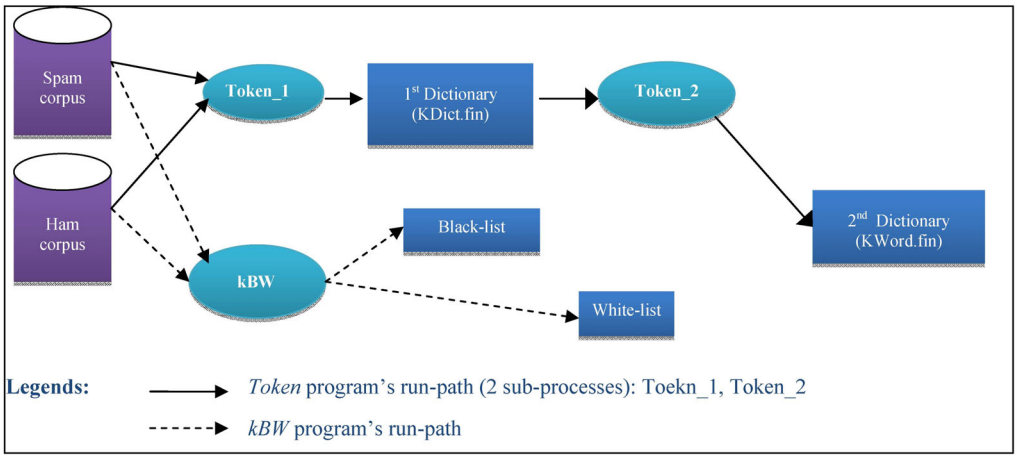


Figure 2. Tokenizing (preparation) phase diagram

The Retrieval phase mainly uses the kCheck program to calculate the likelihood that a user's email is a spam. This phase also uses both webpages as communicating terminal with the user, where the webpage transfers user's email input to the kCheck program and displays the result back to the user. This program will tokenize the input email, select the most popular fifteen words, read their spam-value from the dictionary file, and then apply the overall-formula (in section 3) to get the overall probabilistic value of the input email. If this overall value is greater than 0.9 (90%), the input email is considered as spam. Otherwise (value less than 0.9), it is regarded as normal (legitimate) email. Figure 3 shows the retrieval phase.

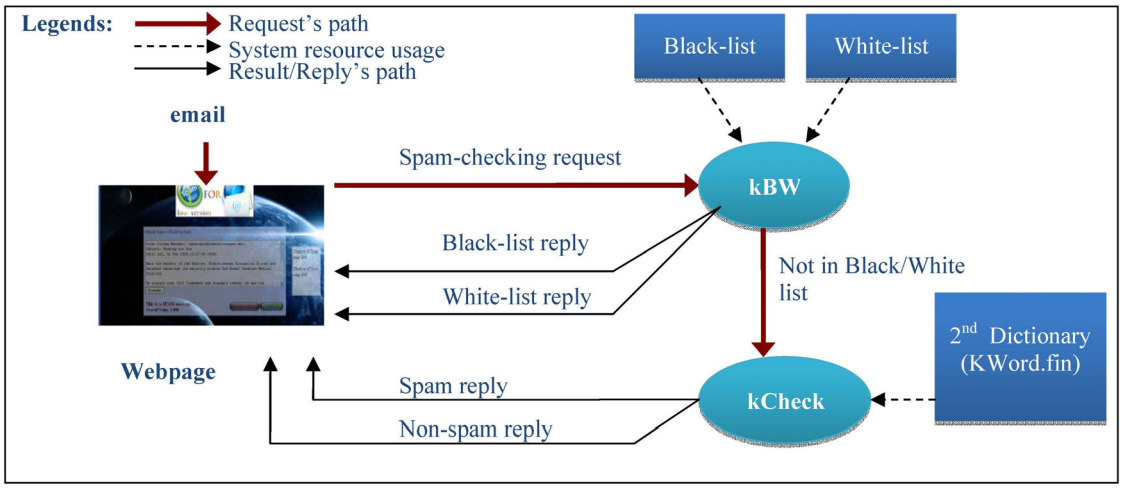


Figure 3. Retrieve (checking) phase diagram

5.2 Results

A corpus of 400 spam emails and 400 ham emails was used in the learning phase, which was extracted from the Apache SpamAssassin Project [23]. The testing process in this model was performed manually to get the feedback from users. Therefore, the testing corpus, which is 200 emails (100 ham and 100 spam), was relatively small.

During the tokenizing (preparation) process, the program produced one Black-list, one White-list, and two wordlist dictionary files. The contents of the body and the header of emails were used in the learning and the testing phase. To test the program, 200 emails were used. The testing sample was split into five groups, and each group contained 40 emails: 20 hams and 20 spams. The testing phase was performed in three phases. In the first phase, a *Static Bayesian Spam Filter (SBSF)* was tested. In this model, the spam tokens and the black and white lists were not updated. The purpose of this stage is to show the difference in performance between the static model and the adaptive model. In the second phase, a *Light Adaptive Bayesian Spam Filter (LABSF)* was tested using the same corpus used in the first phase. LABSF learns from incoming emails and updates the Spam Database engine that contains the tokens of spams. However, this model was tested without using black and white lists. The goal of this phase is to show the advantage of constructing a customized Bayesian Spam Filter for each user and how this approach improves the spam filtering process. In the final phase, the *Enhanced Adaptive Bayesian Spam Filter (EABSF)* was tested. The EABSF is a combination of LABSF and black and white lists. The purpose of this phase is to show what improvement to the LABSF the black and white lists add. Actually, using these lists improves the accuracy of the spam filter by reducing the number of false positives and false negatives. This result is achieved since using white and black lists reduces the number of emails filtered by the adaptive Bayesian spam filter. In other words, the known spam emails or known ham emails according to the black lists and the white lists of a user are classified and removed earlier by the system. Thus, there is no probability to classify them mistakenly by the Bayesian spam filter. Moreover, using these lists reduces the time of the filtering process since known emails according to these lists are not tested using the filter.

Figure 4 shows the results of false positives produced in the three phases. As shown, in some testing groups, the number of false positives produced by three models is the same (group 2). However, in other testing groups (groups 1, 3, 4, and 5) the performance of the SBSF was the worst. Moreover, the performance of the LABSF and EABSF was similar in 3 groups out of 5. Nonetheless, the performance of the EABSF was the best among the three models regarding the produced false positives.

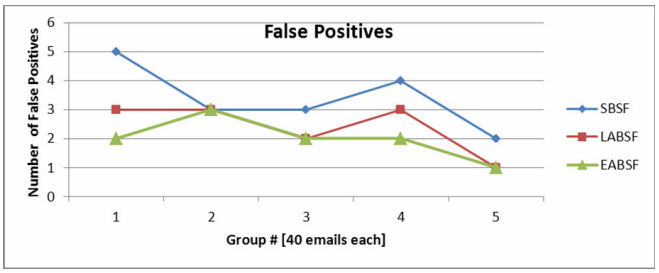


Figure 4. False positives for the three models

Figure 5 compares the number of false negatives produced by three models. As shown in the figure, the worst performance is by the SBSF, while the best performance was achieved by the EABSF. Obviously, adding the learning feature to the Bayesian spam filter increased its performance. Furthermore, combining the learning with the black and white lists has boosted the performance and increased the accuracy in EABSF.

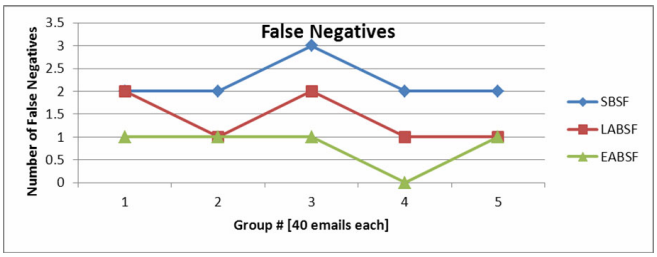


Figure 5. False negatives for the three models

Figure 6 shows the overall recall and precision of the three models. As shown in the figure, the performance of the Bayesian Spam Filter increases as the user feedback and black and white lists are added. That is, the EABSF has the best performance among all models with a recall value of 95% and a precision of 97.9%.

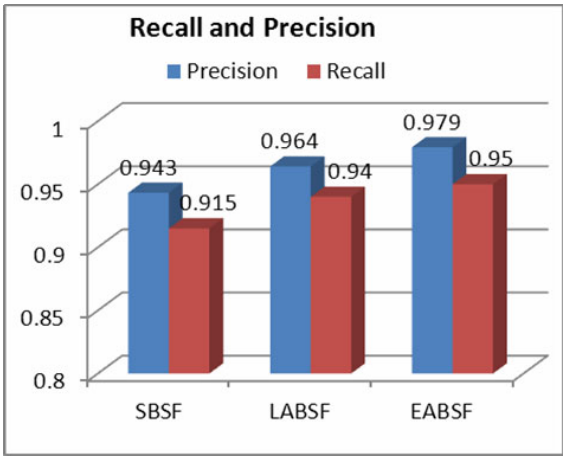


Figure 6. The recall and precision values of the three models

5 Conclusions

This paper has shown the advantages of using a customized Bayesian spam filter for each user. The model in this paper uses a combination of adaptive Bayesian classifier, black lists and white lists. It updates the spam tokens depending on users' feedback so that it builds the spam classifier depending on the precise definition of the spam of each individual. In addition, it updates the white lists and the black lists for each user depending on his/her feedback as well, which increases the accuracy of the spam filter. The paper has introduced an algorithm that shows how the model works in details. In addition, the paper has demonstrated experiments that compare three models of Bayesian spam filters, which are *Static Bayesian Spam Filter*, *Light Adaptive Bayesian Spam Filter*, and *Enhanced Adaptive Bayesian Spam Filter*. The paper has shown that the *Enhanced Adaptive Bayesian Spam Filter* has the best performance among all models.

References

[1] G. Schryen, *Anti-spam Measures: Analysis and Design*, Springer, 2007.

[2] J. Goodman, G. V. Cormack, D. Heckerman, Spam and the Ongoing Battle for the Inbox, *Communications of the ACM*, Vol. 50, No. 2, pp. 24-33, February, 2007.

[3] F. D. Garcia, J.-H. Hoepman, J. van Nieuwenhuizen, Spam Filter Analysis, *Proceedings of the 19th IFIP International Information Security Conference*, Toulouse, France, 2004, pp. 395-410.

[4] D. Cook, J. Hartnett, K. Manderson, J. Scanlan, Catching Spam before It Arrives: Domain Specific Dynamic Blacklists, *Proceedings of the 2006 Australasian Workshops on Grid Computing and E-research*, Hobart, Australia, 2006, pp. 193-202.

[5] J. Golbeck, J. Hendler, Reputation Network Analysis for Email Filtering, *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, 2004, pp. 54-58.

[6] P. Pantel, D. Lin, SpamCop: A Spam Classification and Organization Program, *AAAI-98 Workshop on Learning for Text Categorization*, Madison, WI, 1998, pp. 95-98.

[7] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A Bayesian Approach to Filtering Junk E-Mail, *AAAI Workshop on Learning for Text Categorization*, Madison, WI, 1998, pp. 55-62.

[8] P. Graham, A Plan for Spam, http://www.paulgraham.com/spam.html.

[9] J. S. Kong, B. A. Rezaei, N. Sarshar, V. P. Roychowdhury, P. O. Boykin, Collaborative SPAM Filtering Using E-Mail Networks, *IEEE Computer*, Vol. 39, No. 8, pp. 67-73, August, 2006.

[10] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, P2P-Based Collaborative Spam Detection and Filtering, *Proceedings of the 4th International Conference of Peer-to-Peer Computing*, Zurich, Switzerland, 2004, pp. 1-8.

[11] F. Sebastiani, Machine Learning in Automated Text Categorization, *ACM Computing Surveys*, Vol. 34, No. 1, pp. 1-47, March, 2002.

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

[13] V. P. Deshpande, R. F. Erbacher, C. Harris, An Evaluation of Naïve Bayesian Anti-Spam Filtering Techniques, *Proceedings of the IEEE SMC Workshop on Information Assurance and Security*, West Point, NY, 2007, pp. 333-340.

[14] G. V. Cormack, A. Bratko, Batch and Online Spam Filter Comparison, *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, 2006.

[15] A. Niimi, H. Inomata, M. Miyamoto, O. Konishi, Evaluation of Bayesian Spam Filter and SVM Spam Filter, *Proceedings of the 2nd International Conference on Soft Computing and Intelligent Systems*, Okayama, Japan, 2004, pp. 1-5.

[16] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, C. D. Spyropoulos, An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, 2000, pp. 160-167.

[17] S. Zelikovitz, H. Hirsh, Improving Short-Text Classification Using Unlabeled Data for Classification Problems, *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, CA, 2000, pp. 1191-1198.

[18] M. Healy, S. J. Delany, A. Zamolotskikh, An Assessment of Case-based Reasoning for Short Text Message Classification, *Proceedings of the 15th Irish Conference on Artificial Intelligence and Cognitive Science*, Castlebar, Ireland, 2004, pp. 9-18.

[19] T. Mahmoud, A. M. Mahfouz, SMS Spam Filtering Technique based on Artificial Immune System, *International Journal of Computer Science*, Vo. 9, No. 2, pp. 589-597, March, 2012.

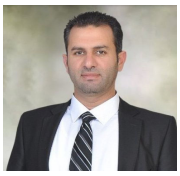
[20] V. Metsis, I. Androutsopoulos, G. Paliouras. Spam Filtering with Naive Bayes – Which Naive Bayes?, *Proceedings of the 3rd Conference on Email and Anti-Spam,* Mountain View, CA, 2006, pp. 1-9.

[21] L. Liu, W. Wei, X. Li, Y. Pan, H. Song, Visual Attention Model Based on Particle Filter, *Proceedings of the KSII Transactions on Internet and Information Systems*, Vol. 10, No. 8, pp. 3791-3805, August, 2016.

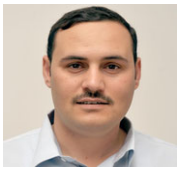
[22] W. Chin, D. Ward, and A. Constantinides. Semi-blind MIMO Channel Tracking using Auxiliary Particle Filtering, *Global Telecommunications Conference*, Taipei, Taiwan, 2002, pp. 322-325.

[23] The Apache Spam Assassin Project, http://spamassassin.apache.org/publiccorpus/.

Biographies



Qussai Yaseen received his Ph.D. in Computer Science from the University of Arkansas at Fayetteville, AR, USA in 2012. In 2014, Dr. Yaseen joined Jordan University of Science and Technology. Currently, he is working on Fog and Cloud Security, IoT security, Wireless Sensor Networks Security, and Insider Threat Mitigation.



Yaser Jararweh received his Ph.D. in Computer Engineering from the University of Arizona in 2010. He is currently an associate professor of computer sciences at Jordan University of Science and Technology. He has co-authored several papers in established journals in fields related to cloud computing, security and Big Data.

Khanh Nguyen Viet got his maser degree in computer science from University of Arkansas at Fayetteville, USA in 2010. He is currently a Software Developer at Epic Corporation at Madison, WI, USA.