A scalable intelligent non-content-based spam-filtering framework

Yong Hua, Ce Guo, E.W.T. Ngai, Mei Liu, Shifeng Chen

1. Introduction

Spam, also known as unsolicited bulk email (UBE), is delivered to millions of recipients every day, occupying network resources and affecting legitimate email traffic. Spam levels in many places of the world have experienced a significant increase; it is estimated that spam accounted for 85% of email traffic in 2007 (Jamali & Geng, 2008). A worldwide cost estimate of spam in terms of lost productivity and IT infrastructure investment stood at well over US$10 billion in 2005 (Jennings, 2005).

One effective way to stop spam from dispatching is to deploy a spam-filtering software system on email servers. In recent years, quite a few anti-spamming approaches have been proposed and applied. Current anti-spamming techniques usually filter spam by analyzing body messages. These methods are shown to be accurate on certain data sets with accuracy as high as 98.8% (Hershkop & Stolfo, 2004). However, body-message-based spam detectors may experience difficulties in the following three scenarios when deployed on a heavily burdened mail server:

1. Manipulation of lexical patterns: Spam may hide behind Hyper-text Markup Language (HTML) or use tricks such as inserting special characters between letters (e.g., “money” may become “m#o%n#$efy” in a spam email). Lexical patterns may easily be altered without losing information intended to be presented to the email recipients. Spammers may employ programs that can change the lexical patterns of spam messages to deceive the text-based filters.

2. Efficiency: A text-based spam filter requires significant processing costs for machines. If such a filter is deployed on the mail client of a personal computer, processing time may be ignored due to small number of emails involved. However, for an email server that processes millions of emails every day, a text-based spam filter may cause fatal problems (i.e., system crashes).

3. Future trends: The number of emails sent and received every day will continue to increase rapidly in the future. Large amounts of resources are needed to enhance the hardware of text-based spam-filtering systems, which in turn leads to significant costs.

For the problems discussed above, an efficient and scalable spam detector is urgently needed. To solve this problem, we propose an Intelligent Hybrid Spam-Filtering Framework (IHSFF) that detects spam by analyzing only five features in email headers, namely, “originator field”, “destination field”, “X-Mailer field”, “sender server IP address” and “mail subject”. Email subjects are digitalized using an algorithm based on n-grams for better performance. Moreover, using real-world data from a well-known ESP in China, we employ various machine-learning algorithms to test the model. Experimental results show that the framework using the Random Forest algorithm achieves good accuracy, recall, precision, and F-measure. With the addition of MetaCost framework, the model works stably well and incurs small costs in various cost-sensitive scenarios.
“sender server IP address” and “mail subject”. Email subjects are digitalized using an algorithm based on n-grams (Cavnar & Trenkle, 1994) for better performance. The framework judges spam without peeking at the mail contents.

The IHSFF has two main advantages. First, it requires less system resources, which makes it extremely suitable for heavily burdened email servers. Second, the system can easily be rescaled and may be deployed alone or with other spam filters. We test our proposed framework with real data from a well-known email service provider (ESP) in China. Various machine-learning algorithms are explored, and the Random Forest (RF) algorithm clearly outperforms others with an accuracy rate higher than 96%. In practice, to most users discarding a legitimate email is much worse than classifying a spam message as legitimate. Our system also demonstrates excellent performance in various cost-sensitive scenarios. In the three cost-sensitive scenarios we considered, little total cost is generated with the proper classification algorithm.

The remainder of this paper is structured as follows. Section 2 reviews the traditional technology applied with anti-spamming and describes related work. Section 3 is a brief introduction to the main methodology of our framework, including the Random Forest decision tree algorithm and the MetaCost algorithm. Section 4 describes our approach. The five features we selected and the digitalizing methods are described in detail. Section 5 contains the evaluation results of our proposed framework. Finally, we discuss the conclusions of the study and future work.

2. Background

Currently two major types of spam-filtering systems are in use, namely, non-machine learning and machine learning. Heuristics, signatures, and blacklisting are typical non-machine learning methods. Although these methods may achieve a high degree of accuracy, they may lead to other significant problems. For example, it is dangerous to deploy a blacklisting filter as a stand-alone filtering system because it has a high rate of classifying legitimate emails as spam (Snyder, 2004).

To solve the problems of non-machine-learning-based spam filters, a large number of studies have been carried out on machine-learning methods. Machine-learning-based spam filters may be further classified into two types, namely “content-based” and “non-content-based”.


Although some of the studies on content-based methods have been fruitful, judging spam by analyzing text may cause some challenging problems. For instance, Georgiou, Dikaiakos, and Stassopoulos (2008) suggested that the widely used HTML technology creates troubles for content-based spam filters. To try to avoid these problems, components other than the contents of email messages have been carefully studied, and non-content-based approaches have been proposed. Lai (2007) empirically studied three different kinds of machine-learning-based spam-filtering algorithms. This research showed that filtering spam using only header information achieved the best performance. A number of behavior models have also been proposed on non-content information in emails. Yue, Abraham, Chi, Hao, and Mo (2007) employed behavior-based features in an artificial immune system for spam filtering. The features they used focus on the sender’s Internet Protocol (IP) addresses. Gomes, Bazuta, Almeida, Almeida, and Meira (2007) analyzed email behaviors between spam and legitimate emails in the aspect of workload. The research aimed to identify and quantify the characteristics that significantly distinguish spam email traffic from legitimate email traffic.

Among the non-content information of emails, header sections are significantly informative. In recent years, spam filters based on email headers have begun to emerge and to achieve acceptable performance. Wang and Chen (2007) presented a statistical analysis of the header section messages of spam and legitimate emails, and explored the possibility of utilizing these messages to perform spam filtering. This approach is particularly challenging, which is of little interest to most users, also provides further information for anti-spamming purposes. Wu (2009) employed back-propagation neural networks for spam filtering. The research used spam behaviors as features to describe an email. The behaviors were digitalized using a rule-based preprocessing method.

Although the field of spam filtering has been well researched, studies on the email filters specified for heavily burdened email servers that consider both efficiency and scalability have yet to be done. This is probably because of a lack of collaboration between researchers and ESPs, which may consequently lead to the following two limitations. First, researchers tend to pay little attention to the heavily burdened servers of ESPs. Second, it is very difficult for researchers to gain access to real-world data from those ESPs.

Unlike most current studies, the purpose of our research is to provide a better solution for the heavily burdened servers of ESPs, where in addition to filtering quality, both efficiency and scalability should be considered. We are fortunate to receive support from a well-known ESP in China, and we have been able to conduct experiments with real-world datasets, thus making our results more convincing.

3. Research methodologies

We employ the Random Forest algorithm as the main classifier, and the MetaCost framework to make the classifier cost-sensitive. In this section, we briefly present the Random Forest algorithm and the MetaCost framework.

3.1. Random Forest

Breiman (2001) introduced Random Forest, which is a combination of tree predictors. Each tree is planted depending on the values of an independently sampled vector. To classify a new object, each tree in the forest gives a classification result (i.e. the tree “votes” for that class). The forest finally chooses the class having the most votes.

Random Forest is fast and does not overfit. It has been proved to be accurate for a number of datasets. Moreover, it can balance errors in class-imbalanced data sets, which makes it suitable for the real-world training data of emails where there are more spam emails than legitimate ones.

3.2. MetaCost framework

Androutsopoulos et al. (2000) suggested that spam filtering is a cost-sensitive process, since blocking a legitimate email is
considered a bigger error than not blocking a spam message. To make the classifier cost-sensitive, we employed MetaCost as a cost-sensitive framework. The MetaCost algorithm proposed by Domingos (1999) is applicable to any number of classes and arbitrary cost matrices.

For a given example x, if the probability of each class j, \( P(j|x) \), is known, the Bayes optimal prediction for x will be the class i that minimizes the conditional risk, which is defined as

\[
R(i|x) = \sum P(j|x) \cdot C(i,j)
\]

where \( C(i,j) \) is the cost when an object belonging to class j is misclassified as class i. The conditional risk \( R(i|x) \) is the expected cost of predicting that x belongs to class i. Given a cost matrix C, the MetaCost algorithm relabels the input samples according to the above definition of conditional risk before the classifier is trained. MetaCost can be utilized to optimize a specified classifier in order to achieve lower total cost.

4. Proposed approach

Spam is often sent via the Internet to sell products or services to email users. For this purpose, spammers send spam emails repeatedly and massively so that their advertisements can be broadly dispatched all over the Internet. To avoid being detected, spam emails often conceal themselves as legitimate ones. Their cover-up techniques later become spam behaviors. For instance, observe the mail header in Fig. 1: spammers have filled the subject line with sequences; therefore, emails from these regions have a relatively low probability of being spam. On the other hand, for regions that are able to take advantage of this feature in judging spam emails.

4.1. Feature selection

We extracted five key features, namely “originator field”, “destination field”, “X-Mailer field”, “sender server IP address” and “mail subject”, to build a vector for a machine-learning-based classifier. Other features may be added according to the requirements of users. The main goal of this section is to design an efficient model for a spam-filtering system specifically for heavily burdened email servers; thus, we must consider the model complexity. To keep the model simple and efficient, we decided to extract and digitalize only five key features from the email header sections, and then employ machine-learning algorithms for classification.

4.1.1. Originator field

Most emails are transmitted across the Internet using the Simple Mail Transfer Protocol (SMTP), which is an Internet standard first defined in RFC2821 and RFC2822. According to RFC2822, the originator field of a message consists of “from”, “sender” (if applicable), and optionally “reply-to” fields. The “from” field may contain one or more mailboxes corresponding to the author(s) of the email. If the “from” field contains only one mailbox, then the “sender” field may be omitted and the only mailbox in the “from” field is treated as the actual sender of the email. If the “from” field contains two or more mailboxes, then the “sender” field, which contains a single mailbox specification, must appear in the message to identify the actual sender of the email.

The SMTP protocol, established in 1982, was designed in a way that allows anybody to use an SMTP server to send emails without authenticating the author’s identity. The information in the originator field is not always legitimate, since spammers may easily use a fake or even null email address to send spam. This action leads to confusion for some black-list-based spam filters, but we are able to take advantage of this feature in judging spam emails.

4.1.2. Destination field

Each email must have at least one destination address so that it can properly arrive at the destination server. The destination field of a message specifies the recipients of the message, and consists of three possible subfields: “To”, “CC”, and “BCC”. Each subfield may have one or more addresses, and each address indicates the intended recipient(s) of the message.

Spammers rarely list the addresses of recipients in the “To” and “CC” fields, since writing a large number of addresses in such fields is considered a strange behavior that may be detected by existing anti-spam systems. Instead these spammers tend to write a long target list in the “BCC” field, which has no limit on the number of recipients. It is estimated that a typical legitimate email has fewer recipients than a spam message, because spam emails are usually used for broadcast. The content analysis provided by Wang and Chen (2007) showed that 84.64% of the legitimate emails contained the addresses of the receivers in the “To” or “CC” field, while 44.26% of the spam emails left nothing in those fields.

4.1.3. X-Mailer field

In the header section, the X-Mailer is an optional field that indicates what email client program or mail user agent (MUA) was used to generate the email. Most spamming programs try to hide themselves by providing invalid X-Mailer fields. It is estimated that most legitimate emails are associated with a legitimate X-Mailer field, while spam emails leaving them out or use randomly generated characters (Wang & Chen, 2007). Consequently, the X-Mailer field can be a sign of spam.

4.1.4. Sender server IP address

It is commonly known that IP addresses may reflect geographic location. The amount and percentage of spam differs from region to region. For example, in some regions, spam can have legal consequences; therefore, emails from these regions have a relatively low probability of being spam. On the other hand, for regions that...
4.1.5. Mail subject

An email subject may provide important evidence in judging an email as spam. A person may even easily judge a message as spam by a glimpse of the subject in some cases. In this research, we propose a simple and effective n-gram-based text processing method to deal with such pieces of natural language information. In this method, each email gets a “subject spam mark.” An email with a higher mark is more likely to be spam.

An n-gram is an n-character slice of a longer string. Although in other literature the term can include the notion of any co-occurring set of characters in a string, in this paper we use the term to describe contiguous slices only. For example, the English word “newspaper” has the following 4-grams: “news”, “ewsp”, “wspa”, “spap”, “pape”, and “aper.” When filtering spam emails, two n-gram frequency profiles are needed. One profile is generated based on spam subjects, serving as a Spam Subject Feature Profile (SSFP). The other profile is generated with legitimate email subjects, forming a Legitimate Email Subject Feature Profile (LESFP).

Let $S$ be a set of email subjects. An n-gram frequency profile based on $S$ is generated using the algorithm below (see Fig. 2). A hash table is employed in this algorithm to temporarily store the n-grams. Each element in the hash table is associated with a count for further processing.

Spammers tend to insert special characters and stop words in the subject (for example, “rich” becomes “r+i+t” in some spam emails) in order to deceive the content-based filters. Hence, special characters and stop words (such as “,” “.”, “,”) are removed before gathering statistics. The n-grams with a higher rank in frequency rank list are considered to be stronger features. n-Grams that appear fewer than three times are not considered as features and are excluded from the profile. We build SSFP and LESFP using a number of spam and normal email subjects respectively, and mark email subjects according to the two profiles. In an n-gram feature profile $P$, each n-gram $g \in P$ has an index position, $rank(g, P)$ (i.e. the rank in the frequency list in $P$). The preceding n-grams in the SSFP are more likely to be a feature of spam, while the preceding n-grams in the LESFP are more likely to be a feature of legitimate email.

For a specified subject $s$, let $G(s) = \{g_1, g_2, g_3, \ldots, g_n\}$ be the set of all n-grams in $s$. When we try to decide how likely an n-gram is to be a feature of the data set that generates $P$, those n-grams that are in $G(s)$ but not in $P$ provide no information. As a result, for a given frequency profile $P$, we consider only the following subset of $G(s)$:

$$VG(s, P) = \{g \in G(s) \land g \in P\}$$

If $VG(s, P)$ is not an empty set, then the distance from subject $s$ to n-gram frequency profile $P$ is defined as the average rank of all n-grams in $VG(s, P)$. For a non-empty $VG(s, P)$, we may perform the following algorithm as shown in Fig. 3 to calculate the distance from subject $s$ to an n-gram frequency profile $P$.

If both $VG(s, SSFP)$ and $VG(s, LESFP)$ are not empty, the values of $distance(s, SSFP)$ and $distance(s, LESFP)$ can both be calculated using the above algorithm. We then define the “spam mark” of an email $e$ as

$$smrk(e) = \frac{\text{distance}(s, LESFP)}{\text{distance}(s, SSFP) + \text{distance}(s, LESFP)}$$

where $s$ is the subject of email $e$. According to this formula, the mark will be no less than 0 and no more than 1. The formula also shows that a specified subject with a higher mark is more likely to be the subject of spam.

The distance algorithm works only for the case when both $VG(s, SSFP)$ and $VG(s, LESFP)$ are not empty. However, this is not always the case. For a given subject $s$, there are a total of three situations where we may encounter empty sets:

Situation 1. $VG(s, SSFP)$ is empty; $VG(s, LESFP)$ is not empty.
Situation 2. $VG(s, SSFP)$ is not empty; $VG(s, LESFP)$ is empty.
Situation 3. $VG(s, SSFP)$ is empty; $VG(s, LESFP)$ is also empty.

In Situation 1 and Situation 2, either $distance(s, SSFP)$ or $distance(s, LESFP)$ cannot be calculated by the distance algorithm. Take Situation 1 for example. The emptiness of $VG(s, SSFP)$ means that there is no evidence showing $s$ is likely to be the subject of spam. However, $VG(s, LESFP)$ is not empty, meaning there are some cues for judging it as legitimate email. As a result, we directly let $score(s) = 0$. Similarly, for those situations in Situation 2, we directly let $score(s) = 1$. In Situation 3, there being no hint from either side, we let $score(s) = 0.5$ for this case.

As discussed above, for all situations, the “spam subject mark” of an email $e$, represented as $smrk(e)$, may be calculated using the following algorithm (see Fig. 4).

4.2. System architecture

To build our Intelligent Hybrid Spam-Filtering Framework, we need a set of legitimate email subjects and a set of spam subjects

---

**Fig. 2.** Frequency profile-generating algorithm for n-grams.

**Fig. 3.** Distance-calculating algorithm.

**Fig. 4.** Subject-marking algorithm.
to generate the two n-gram frequency profiles. When the two profiles are generated, the n-gram-based marking algorithm is ready to mark subjects according to the profiles. Fig. 5 shows the way the system works.

As Fig. 5 shows five pieces of information are extracted from email e. We digitalize them as follows:

- $sender(e)$ is defined to represent whether the sender of email $e$ is valid. The function returns 1 if the address is legitimate; otherwise, it returns 0.
- $nrcpt(e)$ is defined to represent the number of recipients of email $e$.
- $xmalr(e)$ is defined to represent whether email $e$ has a legitimate X-Mailer field. Let $xmalr(e) = 1$ when a legitimate X-Mailer field can be extracted from the header of email $e$; otherwise, we define $xmalr(e) = 0$.
- $ip(e)$ is a normalized value of the sender’s IP address. Each IP address contains 32 binary bits, which can be considered a 32-bit unsigned integer. For email $e$, we extract the sender server’s IP address, consider it a 32-bit unsigned integer $ui_{ip(e)}$, and then normalize the sender IP into the range between 0 to 1, since the values of most other attributes fall within such a range. The normalization is performed this way:
  \[
  ip(e) = \frac{ui_{ip(e)}}{2^{32} - 1}
  \]
- $smrk(e)$ is the return value of the subject-marking algorithm mentioned in Section 4.1.5.

Vector $v(e) = (sender(e), nrcpt(e), xmalr(e), ip(e), smrk(e))$ is then passed to the machine-learning-based classifier for spam detection.

5. Results and discussion

5.1. Datasets and performance measures

The purpose of this research is to provide an efficient and scalable spam-filtering framework for servers of ESPs. To this end, we utilize a real-world database of emails instead of experimental datasets. The real-world database provides a more accurate and updated reflection of actual circumstances. In addition, typical public datasets do not contain information on some required features for the proposed method. Therefore, the results of this study are not comparable to those of other studies.

The datasets we used are obtained from a well-known Chinese portal website, which has more than 45 million registered email users. Datasets extracted from the email servers include records of more than 30,000 customers. We use two datasets for building and estimating the model. One data contains a set of email subjects with 33,209 samples, each of which has a class label. This data set is employed to generate the two profiles (SSFP, LESFP) for the subject-marking algorithm. The other dataset has 21,725 integrated email headers with class labels. This dataset is utilized to train and test the proposed model.

In this research, legitimate emails are regarded as positive samples, while spam emails are regarded as negative ones. In the
following discussion, TP stands for the number of true positive samples; TN stands for the number of true negative samples, while FP and FN stand for the number of false positive and false negative samples, respectively.

To evaluate the model’s performance, we calculated four measures: accuracy, precision, recall, and the F-measure. Accuracy is the most commonly used evaluation criterion that measures the degree of veracity. It is formally defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$ (5)

The three other measures – precision, recall, and F-measure – are popular criteria in text-categorizing tasks (Koprinska, Poon, Clark, & Chan, 2007). Precision is closely related to accuracy, which is the percentage of correct predictions. Recall in binary classification is also called “sensitivity” and examines the probability of a true positive sample being retrieved. Because a trade-off between precision and recall usually exists, it is crucial to find a balance between the two measures. The F-measure or F-score in statistics considers both measures, which can be interpreted as a weighted average of precision and recall. These three measures are formally defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$ (6)

$$\text{Recall} = \frac{TP}{TP + FN}$$ (7)

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$ (8)

In practice, most users discarding a legitimate email is much worse than classifying a spam message as legitimate. This implies that, if legitimate emails are treated as positive samples, a high recall is particularly important in spam filtering. Therefore, we have to introduce cost-sensitive scenarios to this research. A false negative error is considered \(k\) times more costly than a false positive error.

To make the accuracy sensitive to this cost, we define “cost” as follows:

$$\text{cost} = k \cdot FN + FP$$ (9)

We examined three cost scenarios: \(k = 1\), \(k = 9\), and \(k = 999\) (Androutsopoulos et al., 2000). For the cost-sensitive scale of \(k = 1\), no cost was considered. For \(k = 9\), discarding a legitimate email was considered nine times more harmful than classifying a spam message as legitimate. When \(k = 999\), discarding a legitimate email was considered an unforgivable action.

5.2. Experimental results

In this research, we selected five popular classification algorithms, namely, Random Forest (RF), Decision Tree C4.5, Naive Bayesian (NB), Bayesian Network (BN), and Support Vector Machine (SVM), for estimating and comparing. For SVM, we tested four different kernel functions: linear, polynomial, radial basis, and sigmoid functions.

In some studies, data are artificially divided into training sets and testing sets. The results may be badly influenced by testing hypotheses suggested by the data. To eliminate such influences, we performed a 5-fold cross-validation in the experiment.

We partitioned the original sample into five subsets and repeated the cross-validation process five times, using each of the five subsets exactly once as the validation data. We then averaged the five results from the folds to produce the final estimation.

To estimate the basic performance of these algorithms, we calculated accuracy, recall, precision, and the F-Measure, as shown in Table 1. The numbers in parentheses are the standard deviations.

For the convenience of comparing the algorithms in a specified cost-sensitive scenario, their performances are shown in Fig. 6a–c. In the no cost situation, RF clearly outperforms all other classifiers in terms of accuracy, precision, and the F-measure (Fig. 6a). The values of the four measures are in the range between 0.929 and 0.968. NB has the worst accuracy, below 0.7. SVM with sigmoid kernel function performs the worst in recall and the F-measure.

When we punish the misclassification of legitimate emails as spam with more cost (\(k = 9\)), RF clearly remained the best performing classifier in terms of accuracy, precision, and the F-measure (Fig. 6b). Although the recall of RF (0.9618) is not the highest, it is only a little lower than the best performing BN classifier (0.9920). On the other hand, the precision of the BN classifier (0.6691) is significantly lower than the precision of the RF (0.8707), which means that BN fails to filter out spam mails in a trade-off for higher recall values.

Finally, in the case of \(k = 999\), RF has the highest accuracy (0.9219) and F-measure (0.8593). Its precision (0.7626) and recall (0.9842) are only slightly lower than the best values (0.8072 and 0.9762).

Table 1: Performances of different machine-learning classifiers.

<table>
<thead>
<tr>
<th>(k)</th>
<th>RF</th>
<th>C4.5</th>
<th>NB</th>
<th>BN</th>
<th>SVM1</th>
<th>SVM2</th>
<th>SVM3</th>
<th>SVM4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9674(0.0012)</td>
<td>0.9601(0.0040)</td>
<td>0.8826(0.0037)</td>
<td>0.9057(0.0028)</td>
<td>0.9015(0.0075)</td>
<td>0.8979(0.0070)</td>
<td>0.8962(0.0040)</td>
<td>0.8876(0.0116)</td>
</tr>
<tr>
<td>9</td>
<td>0.9561(0.0055)</td>
<td>0.9402(0.0051)</td>
<td>0.8762(0.0042)</td>
<td>0.8545(0.0069)</td>
<td>0.9031(0.0049)</td>
<td>0.8975(0.0067)</td>
<td>0.8956(0.0042)</td>
<td>0.8829(0.0195)</td>
</tr>
<tr>
<td>999</td>
<td>0.9219(0.0335)</td>
<td>0.7522(0.0045)</td>
<td>0.6750(0.0041)</td>
<td>0.7622(0.0040)</td>
<td>0.9031(0.0049)</td>
<td>0.8975(0.0067)</td>
<td>0.8956(0.0042)</td>
<td>0.8829(0.0195)</td>
</tr>
<tr>
<td>1</td>
<td>0.9353(0.0054)</td>
<td>0.9286(0.0151)</td>
<td>0.6842(0.0064)</td>
<td>0.7307(0.0075)</td>
<td>0.7870(0.0077)</td>
<td>0.7896(0.0165)</td>
<td>0.8116(0.0082)</td>
<td>0.7945(0.0306)</td>
</tr>
<tr>
<td>9</td>
<td>0.8707(0.0152)</td>
<td>0.8226(0.0151)</td>
<td>0.6691(0.0076)</td>
<td>0.6262(0.0114)</td>
<td>0.7799(0.0076)</td>
<td>0.7513(0.0322)</td>
<td>0.8072(0.0053)</td>
<td>0.7616(0.0713)</td>
</tr>
<tr>
<td>999</td>
<td>0.7626(0.0089)</td>
<td>0.4944(0.0046)</td>
<td>0.4937(0.0042)</td>
<td>0.5047(0.0042)</td>
<td>0.7799(0.0076)</td>
<td>0.7513(0.0322)</td>
<td>0.8072(0.0053)</td>
<td>0.7616(0.0713)</td>
</tr>
<tr>
<td>1</td>
<td>0.9299(0.0037)</td>
<td>0.9054(0.0154)</td>
<td>0.9775(0.0066)</td>
<td>0.9675(0.0045)</td>
<td>0.8146(0.0550)</td>
<td>0.8005(0.0500)</td>
<td>0.7443(0.0212)</td>
<td>0.7242(0.0219)</td>
</tr>
<tr>
<td>9</td>
<td>0.9618(0.0056)</td>
<td>0.9605(0.0046)</td>
<td>0.9679(0.0035)</td>
<td>0.9920(0.0025)</td>
<td>0.8361(0.0171)</td>
<td>0.8680(0.0504)</td>
<td>0.7477(0.0211)</td>
<td>0.7688(0.0487)</td>
</tr>
<tr>
<td>999</td>
<td>0.9842(0.0028)</td>
<td>0.9992(0.0008)</td>
<td>0.9875(0.0016)</td>
<td>1.0000(0.0000)</td>
<td>0.8361(0.0171)</td>
<td>0.8680(0.0504)</td>
<td>0.7477(0.0211)</td>
<td>0.7688(0.0487)</td>
</tr>
<tr>
<td>1</td>
<td>0.9326(0.0024)</td>
<td>0.9167(0.0083)</td>
<td>0.7980(0.0060)</td>
<td>0.8325(0.0039)</td>
<td>0.7995(0.0042)</td>
<td>0.7941(0.0034)</td>
<td>0.7763(0.0012)</td>
<td>0.7756(0.0025)</td>
</tr>
<tr>
<td>9</td>
<td>0.9140(0.0102)</td>
<td>0.8862(0.0087)</td>
<td>0.7912(0.0061)</td>
<td>0.7677(0.0080)</td>
<td>0.8069(0.0109)</td>
<td>0.8039(0.0109)</td>
<td>0.7762(0.0117)</td>
<td>0.7621(0.0224)</td>
</tr>
<tr>
<td>999</td>
<td>0.8593(0.0053)</td>
<td>0.6615(0.0042)</td>
<td>0.6608(0.0040)</td>
<td>0.6708(0.0037)</td>
<td>0.8069(0.0109)</td>
<td>0.8039(0.0109)</td>
<td>0.7762(0.0117)</td>
<td>0.7621(0.0224)</td>
</tr>
</tbody>
</table>

RF: Random Forest.
C4.5: C4.5 Decision Tree Algorithms.
NB: Naïve Bayes.
BN: Bayesian Network.
The total cost (TC) is the sum of costs in each fold. The maximum possible total cost (MAXTC) is achieved by incorrectly classifying every instance. For a specified cost-sensitive scenario, the maximum total cost can be calculated as follows:

\[ \text{MAXTC} = \lambda (TP + FN) + (TN + FP) \]  

(10)

The total cost may be normalized into the range \([0 \ldots 1]\) for comparison. For a specific cost-sensitive scenario (i.e., \(\lambda\) is specified), the normalized total cost (NC) is calculated as follows:

\[ \text{NC} = \frac{\text{Total Cost}}{\text{MAXTC}} = \frac{\lambda (TP + FN) + (TN + FP)}{\lambda (TP + FN) + (TN + FP)} \]  

(11)

The values of NC are also shown in Table 2.

Fig. 8 shows the normalized total cost in the three cost-sensitive scenarios.

The MetaCost framework has a different effect on different classification algorithms (Fig. 8). Both C4.5 and the Bayesian-based algorithms tend to be more sensitive to the MetaCost framework. The normalized total costs of these algorithms are not significantly increased, or they may even decrease, with an increase in \(\lambda\). The normalized total costs of NB even decrease dramatically from 0.1174 to 0.0023 as \(\lambda\) increases from 1 to 999. The normalized total cost of RF is well controlled under 0.0401, which is also acceptable. But SVMs perform poorly in the MetaCost framework, leading to the highest cost in each situation. The total cost for RF is consistently low compared with other classifiers for the three cost-sensitive cases.

Through experimental analysis, the RF classifier is proved to be accurate with an acceptable low cost, which is probably because the algorithm is based on a subtle voting system. This stability of the voting system probably makes it insensitive to the MetaCost framework, which leads to a slightly larger cost in the situation when \(\lambda\) reaches 999 compared with the C4.5 and Bayesian-based algorithms. It is likely that the MetaCost framework is quite effective on Bayesian-based classifiers, which makes the classifier perform with less over-blocking. In the situation where \(\lambda = 999\), BN judges legitimate emails carefully and performs no over-blocking, which leads to the smallest cost.

According to the results, we may choose different learning algorithms for different situations. RF has the overall best performance and stability in the various cost-sensitive situations.

The IHSFF is fast because it does not need to touch mail content, and it may accurately filter out most spam emails in a relatively shorter time. At the same time, the framework may be extended. The classifiers may be slightly modified to classify emails into three...
classes, namely "legitimate", "spam", and "ambiguous". Those ambiguous emails may be passed to other classifiers, which are typically slower for further determination. For instance, the Random Forest classifier is taken for classification and 10 trees are planted. An ambiguous sample may be defined as those samples with 4–6 votes of spam. Only those ambiguous samples are needed for further determination, which significantly reduces the total processing time.

6. Concluding remarks and future research

A typical large ESP has to deal with more than a hundred million email messages every day. In addition to a reasonable accuracy
rate, efficiency and scalability should be seriously considered in a spam-filtering system running on these servers. Here, we have proposed a new spam-filtering framework called the Intelligent Hybrid Spam-Filtering Framework (IHSFF) for heavily burdened email servers. The IHSFF does not need to analyze mail content for spam detection. We use sender validity, number of recipients, sender IP address, ESP validity, and mail subject as features, and then use a machine-learning algorithm for spam filtering. Our approach has the following advantages:

1. **High efficiency:** The model is kept simple, and it does not consider mail content; hence, it consumes less system resources, which makes it more efficient than text-based methods. It is especially suitable for deployment in large email servers.

2. **High accuracy and low cost:** With an appropriate classifier, the system achieves accuracy rates higher than 96%. In different cost-sensitive situations, a different classifier may be chosen for lower total cost. In the various cost-sensitive situations we tested, little total cost was generated while using the proper algorithm.

3. **High scalability:** The IHSFF is efficient and accurate. Hence, it may be combined with other filters while maintaining high efficiency. The framework may be rescaled when higher accuracy is needed, and ambiguous emails may be sent to another filter for further determination. A good trade-off between efficiency and performance can be found through experimental methods.

Currently, little of the research on spam filtering focuses on heavily burdened email servers. The datasets used in previous studies are usually public datasets, which may not reflect real-world situations. Unlike most previous studies, the purpose of our research has been to provide a better solution for the heavily burdened servers of ESPs, where efficiency and scalability should be considered in addition to filtering quality. We are fortunate to have the opportunity to cooperate with a well-known ESP in China for the entire study and to test our IHSFF with real-world databases, which makes our experiment results more convincing.

For spam filters on large email servers, much remains to be done to achieve better efficiency, scalability, and accuracy. In future research, our proposed system could be enhanced with the following considerations: First, we did not consider every combination of features in email headers; other features may be added according to user requirements. We selected only five key features from the email header; there may be other ones as well. Adding the right features into the model may contribute to improving accuracy while maintaining efficiency. Second, the subject-marking algorithm may be improved with techniques other than n-grams, or part of the algorithm may be changed to increase accuracy during marking. Third, the digitalization algorithms in this approach may be optimized. Subtle changes in the algorithms may make the system even more efficient and accurate. And fourth, the decision-tree-based classifier Random Forest performed well in the experiment. The decision trees constructed are readily interpretable. Additional knowledge about spam may be discovered by inspecting these trees more closely.

**Acknowledgements**

This research was partly supported by National Natural Science Foundation of China (NSFC, Project No.: 70801020) and The Hong Kong Polytechnic University under a research Grant No. G-YY71 and YY71 and “211 Project” of Guangdong University of Foreign Studies.

**References**


