PAPER *Special Issue on New Technologies and their Applications of the Internet*

# Density-based spam detector

**Kenichi YOSHIDA**[†a)], *Member*, **Fuminori ADACHI**[††b)], **Takashi WASHIO**[††c)],
**Hiroshi MOTODA**[††d)], **Teruaki HOMMA**[†††e)], **Akihiro NAKASHIMA**[†††f)], ***Nonmembers***,
**Hiromitsu FUJIKAWA**[††††g)], ***and*** **Katsuyuki YAMAZAKI**[††††h)], ***Members***

**SUMMARY** The volume of mass unsolicited electronic mail, often known as spam, has recently increased enormously and has become a serious threat not only to the Internet but also to society. This paper proposes a new spam detection method which uses document space density information. Although the proposed method requires extensive e-mail traffic to acquire the necessary information, it can achieve perfect detection (i.e., both recall and precision is 100%) under practical conditions. A direct-mapped cache method contributes to the handling of over 13,000 e-mail messages per second. Experimental results, which were conducted using over 50 million actual e-mail messages, are also reported in this paper.

***key words:*** *spam, unsupervised learning, document space density, direct-mapped cache*

## 1. Introduction

Mass unsolicited electronic mail, often known as spam[1] [*], has recently increased enormously and has become a serious threat not only to the Internet but also to society. This is especially true in Japan, where mobile phones have e-mail capability and e-mail messages through these devices have become indispensable to society. Under these circumstances, there exists a strong requirement for a spam filter which can protect large mail servers. However, none of the currently known spam filters can effectively cope with the huge volume of traffic with sufficient accuracy.

Even though a lot of studies have been undertaken to create and improve spam filters, most of them are for e-mail clients which are used on a terminal. Such spam filters, for e-mail clients, should be accurate, easy to personalize, and easy to use. However, the required characteristics of a spam filter for e-mail servers are slightly different. They are:

- High processing speed:
  Large ISP e-mail servers have to handle billions of e-mail messages per day. This means that the spam filter has to handle more than 1000 e-mail messages per second. Since the most well known spam filter program requires 10 to 100 milliseconds to deal with each e-mail message, performance improvement is necessary.
- Ease of maintenance:
  Most of the traditional spam filtering methods requires maintenance of their databases so that they can handle new types of spam. Unfortunately, spammers tend to be very productive and are always producing new types of spam. This makes maintenance work difficult, especially for e-mail servers. The learning function of the traditional method is adequate for individual customers but not for groups of customers. This also makes maintenance difficult. A maintenance free method such as an unsupervised learning method is desirable.
- High accuracy:
  Although accuracy is important for client spam filters, it is also important for a spam filter to be accurate for e-mail servers. When considering an anti-spam arrangement, the requirements for judgment accuracy are different between clients and servers. A server requires misjudgment probability of normal e-mail message being marked as spam to be zero, whereas that strict requirement is not as necessary for clients. In other words, a method of anti-spam arrangement that achieves the above

Manuscript received March 11, 2004.
Manuscript revised June 1, 2004.
Final manuscript received June 1, 2004.

[†] Graduate School of Business Science, University of Tsukuba Otsuka 3-29-1, Bunkyo, Tokyo 112-0012, Japan

[††] The Institute of Scientific and Industrial Research, Osaka University 8-1, Mihogaoka, Ibarakishi, Osaka 567-0047, Japan

[†††] KDDI Corporation Garden Air Tower, 3-10-10, Iidabashi, Chiyoda, Tokyo 102-8460, Japan

[††††] KDDI R&D Laboratories Inc. 2-1-15 Ohara, Kamifukuoka, Saitama 356-8502, Japan

a) E-mail: yoshida@gssm.otsuka.tsukuba.ac.jp
b) E-mail: adachi@ar.sanken.osaka-u.ac.jp
c) E-mail: washio@sanken.osaka-u.ac.jp
d) E-mail: motoda@sanken.osaka-u.ac.jp
e) E-mail: teruaki@kddi.com
f) E-mail: nakasima@kddi.com
g) E-mail: fujikawa@kddilabs.jp
h) E-mail: yamazaki@kddilabs.jp

[*] Mass electric mail includes both unsolicited mail and solicited mail. In general, mass solicited mail includes mail magazines, error mails, etc., and spam mainly refers to unsolicited mail. In this paper, spam refers to both types of mass mail. Since a short white list, i.e., a database which stores the list of proper senders, seems to work well with the method presented in this paper, we aren't concerned about this confusion of solicited and unsolicited mail. See Section 5.2 for details.

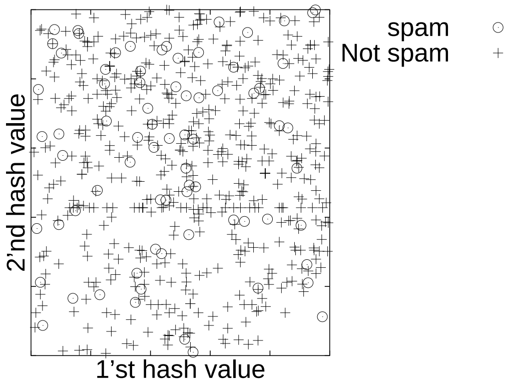


**Fig. 1** Vector Representation of E-mail Message

requirement will only be implemented within a network server environment.

- Privacy protection:
  In order to be implemented on an e-mail server within a network, it is desirable that a method and related operations do not directly look or reveal the content of e-mail. Some type of abstraction needs to be done at the first stage of the method.

This paper reports on a new spam detection method for e-mail servers. Two key ideas of our study are 1) the use of document space density [2] information, and 2) an efficient implementation of the first idea through the use of a direct-mapped cache [3]. The proposed method requires extensive volumes of e-mail traffic to acquire the necessary density information. Thus it is not adequate to use this method for client terminals. However, the latter three characteristics, i.e., ease of maintenance, high accuracy and privacy protection, are achieved. To realize the first characteristics, i.e., high processing speed, an on-line unsupervised learning engine with a direct-mapped cache method is developed.

Section 2 of this paper first surveys related work and determines their limitations in order to clarify the motivation of this research. Section 3 explains the analysis of document space density with a direct-mapped cache engine. It also explain the possible refinements of basic algorithm. Section 4 reports on the experimental results that used over 50 million actual e-mail messages. It also compares our method with other methods, and compares the basic algorithm with possible refinements. Section 5 discusses related topics. Finally, Section 6 concludes our findings.

## 2. Related work

Since spam has become a serious threat to society, a lot of study has been undertaken to create spam filters to protect e-mail users, e.g., [4]–[7], and [8]. Some of them use a Bayesian-like approach [5], [6], or a rule-based approach [7], and some use a checksum database [4], [8] to detect spam.

Vector representation, e.g., TF IDF[2], combined with machine learning techniques [9] are commonly used to detect spam. A fundamental dilemma is the difficulty of the learning problem. Figure 1 shows a sample vector representation of a group of e-mail messages. In Figure 1, X axis is the first hash value of the e-mail text and Y axis is the second hash value (See Section 3 for the details). Spammers today seem to have a great deal of knowledge about techniques used to detect spam. They try to make the size of their information much smaller. For example, they make shorter spam message with minor alterations. They also add random words so that random words disturb the statistical analysis. Such tricks make the learning task difficult. In other words, finding a function to discriminate between spam and non-spam on this representation alone is not a simple task.

We also use related representation (See next section for details). However, to treat Japanese and English e-mail together in an efficient way requires that we choose hash-based text representation. Hash-based text representation is one of the basic text representation methods [10] and it is used for a variety of purposes, e.g., text retrieval [11], text compression [12], and spam filtering[4], [8]. Since hash-based text representation doesn't require a morphological analysis of Japanese text, it therefore improves the performance of our method.

A high speed text search engine is an important component of our method. A direct-mapped cache is the core of our engine. It is a kind of hash table that simply overwrites duplicate entries, and is originally developed as a substitute of the LRU cache to implement cache memory of CPU [3]. LRU performance on heavily maldistributed data is studied in various network applications. For example, [13] analyzes WWW traffic and reveals LRU's high performance on gathering maldistributed WWW data. In our study, the direct-mapped cache [3] is used to gather maldistributed spam messages.

## 3. Density-based spam detector

The analysis of document space density itself and the unsupervised learning engine with a direct-mapped cache are the key ideas of our study. This section explains these two ideas with an implemented system. Possible refinements of memory management are also described.

### 3.1 Document Space Density

Although most of the conventional spam filters use vector representation for the basic representation of data, we use document space density [2] as the key piece of information to distinguish spam messages from other e-

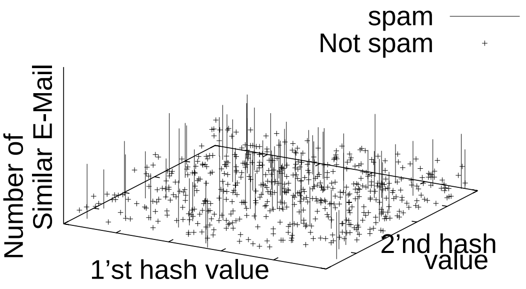


**Fig. 2** Histogram of Similar Mail Messages

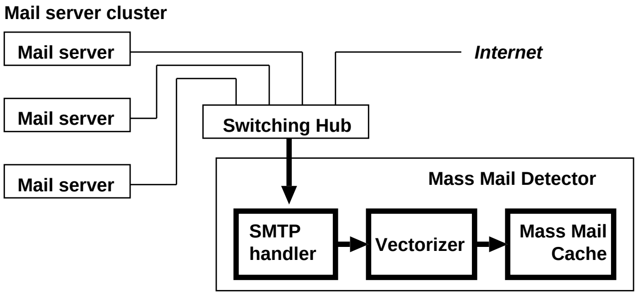


**Fig. 3** Configuration of Mass-Mail Detector

mail messages. More precisely, we just count the number of similar e-mail messages. By counting the number of similar e-mail messages, we can estimate the local document density around the mail message.

Figure 2 shows the histogram of e-mail messages shown in Figure 1. The X and Y axis are that of Figure 1. The Z axis is the number of similar e-mail messages. As clearly shown in the Figure, the use of the histogram makes distinguishing spam from other e-mail far easier. Actually, experimental results reported in Section 4 showed that simple threshold is enough to distinguish spam messages from other e-mail messages.

Spammers conduct marketing, commercial, and even unethical activities by sending out a huge amount of spam. This high volume is required as it is the only way to receive enough economical benefit. There is therefore a heavy maldistribution on e-mail traffic, making document space density a good index to identify spam. Although ordinary users seldom send more than 1000 similar e-mail messages, spammers have to send the same spam far more than that. Note that some of the unethical spam messages are said to be difficult to judge even for a human. However, the existence of over thousand identical e-mail messages makes the fact clear.

### 3.2 System Configuration

Figure 3 shows the system configuration of MMD (Mass-Mail Detector) that we have implemented. By monitoring network packets at the switching hub, the SMTP handler analyzes SMTP traffic between mail servers and reconstructs the text of e-mail messages. Then, Vectorizer transfers the text into vector representation.

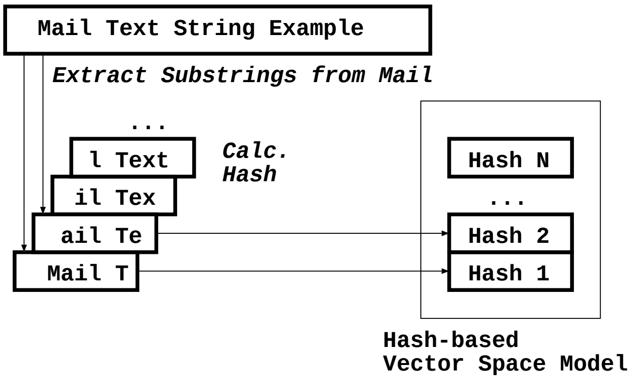


**Fig. 4** Hash-based Vector Representation

There are various vector representations, such as term frequency and N gram. Although they are candidate representation, we use a hash-based vector representation (See Figure 4). From each e-mail message, hash values of each length L substring are calculated,[†] and then the first N of them are used as vector representation of the e-mail message.

Japanese, English and other languages are used in mobile phone e-mail in Japan. Bigram is known to work well for Japanese. Term frequency is commonly used for English. The hash-based representation can extract enough information from various languages in a singular and simplistic way. Its efficiency is also the reason we choose this as our representation method.

### 3.3 Caching Architecture

An unsupervised learning engine is used to find spam from huge volumes of e-mail traffic. Since it has to handle over 1000 e-mail messages per second and it needs to check a million previous e-mail messages to handle the current single e-mail message, naive implementation requires over a billion similarity checks and this isn't realistic. To solve this problem, we have developed a new type of unsupervised learning engine which uses a direct-mapped cache [3] architecture to speed up processing.

Figure 5 shows the data structure and Figure 6 shows the algorithm. The hash database in Figure 5 has M entries. It stores the hash values of each e-mail message and the number of similar e-mail messages. The direct-mapped cache has m entries, and copies the n% of the hash values. It also stores the pointer to the

[†] We use the standard hash function provided in linux C library.

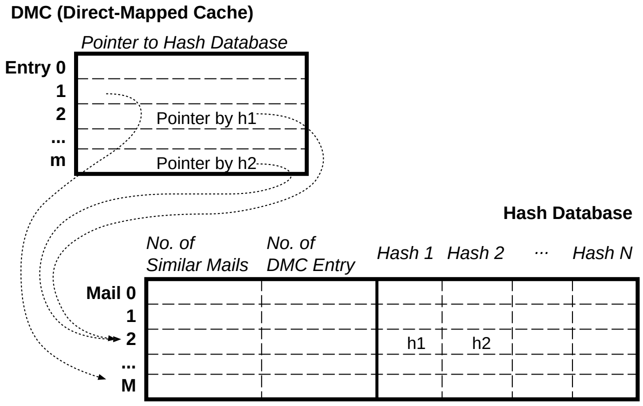


**Fig. 5** Data Structure of Mass Mail Cache

e-mail's entry in the hash database. To check a single piece of e-mail message, in order to find similar previous e-mail message which share S% of the same hash values, the algorithm shown in Figure 6 first checks the direct-mapped cache. The direct-mapped cache is a simple hash table, and the algorithm can find the entries of e-mail messages which have the same hash value through this cache.

More precisely, the algorithm checks the entry of the direct-mapped cache for each hash value of new e-mail message. If the entry of direct-mapped cache points an entry in hash database, the algorithm compare the similarity between the new e-mail message and the stored mail message in the hash database that is pointed by the direct-mapped cache entry.

The contents in the direct-mapped cache are simply overwritten if the hash values are overlapped. When all the hash values in the direct-mapped cache are overwritten by other e-mail messages, the algorithm deletes the entry in the hash database for the overwritten e-mail message so that it can reuse the memory space of the hash database.

### 3.4 Auxiliary Memory Management for Hash Database

Although the mechanism described so far has a basic memory management function, there exists a possibility where the algorithm encounters a lack of free space for new hash database entry. Table 1 shows the possible auxiliary memory management methods that are used in such a case.

RND makes memory space for the new entry by randomly deleting the present hash database entry when the basic memory management mentioned above is not enough. RND2 is a slightly modified version. It retains the entries that are fetched more than once, and entries that are fetched only once will be deleted at random. LRU allocates memory space for the new entry by deleting a least recently used entry. LRU2 also uses a LRU stack. But the entries that are fetched more than 2 times are retained.

| | |
|---|---|
| RND | Basically, the memory space for the hash database entry will be reused by collecting the spaces that are deleted by the procedure Set-DMC-Entry. If this default allocation is not enough, the memory space for the new entry will be allocated by randomly deleting present hash database entries. |
| RND2 | Modified version of RND. The entries that are fetched more than 2 times are retained. Entries that are fetched only once will be deleted at random. |
| LRU | If the default allocation is not enough, memory space for the new entry will be allocated by deleting a least recently used entry. |
| LRU2 | Modified version of LRU. But the entries that are fetched more than 2 times are retained. |

**Table 1** Auxiliary Memory Management Methods

| | |
|---|---|
| Total Number of e-mail message | 53,985,002 |
| Total Number of "spam" | 12,324,762 |
| m: DMC entry | 2,000,000 |
| M: Hash Database entry | 1,000,000 |
| D: "spam" threshold | 100 |
| L: length of substring | 9 |
| N: Number of hash values for each e-mail | 100 |
| n: Percentage of hash values copied in DMC | 10 % |
| S: Similarity threshold | 90 % |
| Memory Size | 825 MByte |
| CPU time | 4340 sec |
| Number of "spam" Type | 14,320 |
| Percentage of "spam" Type | 22.8 % |
| Estimated Recall (See Fig 8) | 100 % |
| Precision | 100 % |

**Table 2** Summary of Experimental Results

The next section compares experiments of these auxiliary memory management methods.

## 4. Experimental Results

Unfortunately, we are not allowed to disclose all of our findings in order to protect privacy. This section only summarizes important statistics.

### 4.1 Results on "spam" through Mobile Phone

We have analyzed actual SMTP traffic transferred through the segments of a genuine mail site. Since no single segment transfers all of the SMTP traffic, we were only able to analyze a part of the overall traffic. A Pentium 4, 2.4 GHz, desktop computer with 2Gbytes of memory was used for the experimentation.

Table 2 summarizes the results. 22.8% of the total number of e-mail messages are spam, and the distribution of similar e-mail message is shown in Figure 7. As seen in various network data, [13] for example, the distribution of similar e-mail messages follows Zipf's law (See Figure 7).

```
Main Procedure Check-Mail
Input
    T: Text of New Mail
Var
    h: Hash value
begin
    New-Hash-DB-Candidate
        ← Make N Hash values from T

    // Compare Hash Values of New Mail T
    //      and Mails stored in Hash Database.
    // Mails in Hash Database
    //      are Searched through DMC.
    for h in New-Hash-DB-Candidate do
        // Entry of h in DMC points
        //      Mail in Hash Database.
        if Similar(Mail in Hash Database
                        pointed by h,
                New-Hash-DB-Candidate)
        then Update-Similar-Mail (Mail in
                Hash Database pointed by h)
            exit Check-Mail

    // If No Similar Entry exists
    //          in Hash Database
    Store-New-Mail(New-Hash-DB-Candidate)
end

Function Similar
Input
    H1: Hash Database entry
    H2: New-Hash-DB-Candidate
begin
    if H1 and H2 share S same hash value
    then return Yes
    else return No
end
```

```
Procedure Update-Similar-Mail
Input
    H1: Hash Database entry
Var
    h: Hash value
begin
    Increment "No.of Similar Mail" of H1
    // Copy n% of hash values from H1 to DMC
    for first n h in H1 do
        Set-DMC-Entry(current h, H1)
    if No.of Similar Mail > D
    then Mark H1 as "spam"
end

Procedure Store-New-Mail
Input
    H2: New-Hash-DB-Candidate
Var
    h: Hash value
begin
    Store H2 as New Hash Database Entry
    // Copy n% of hash values from H2 to DMC
    for first n h in H2 do
        Set-DMC-Entry(current h, H2)
    Set "No.of Similar Mails" as 1
    Set "No.of DMC Entry" as n
end

Procedure Set-DMC-Entry
Input
    h: Hash value
    H: Hash Database Entry
begin
    // If h overwrites old entry,
    //      Decrement "No.of DMC Entry"
    if Previous h already point
                Hash Database Entry: e
        & e ≠ H
    then Decrement "No.of DMC Entry" of e

    // Delete Hash Database entry
    //      pointed by no DMC entry.
    if "No.of DMC Entry" of e = 0
    then Delete e from Hash Database
        & Clear DMC entry which point e

    Set DMC Entry for h so that it points H
end
```

**Fig. 6** Mass Mail Caching Algorithm

The similarity threshold used in the experimentations is 90%. E-mail messages transferred more than 100 times are marked as spam. 100% of marked e-mail messages are mass mail messages as is defined. One million entries for the hash database and two million entries for the direct-mapped cache consumed 825M bytes. 100 substrings whose length being 9 are used to make hash values for each single e-mail message. 10% of the hash values in the hash database are copied in the direct-mapped cache.

Note that the results might include both unsolicited and solicited mass e-mail messages. However removing solicited mass e-mail messages, e.g., mail-based magazines, by using a short white list is not difficult, and we aren't concerned about the confusion between solicited and unsolicited mail. (See Section 5.2 for details). A short white list seems to work well with density based analysis.

Since counting the recall rate directly from actual e-mail traffic has privacy issues and is difficult, we have performed preparatory experimentation. From the traffic mentioned above, the first 10 million e-mail messages are extracted and merged with pseudo spam messages. The preparatory experimentation measures the recall rate over this pseudo spam (See Figure 8).

100 seeds of spam are prepared and each seed is randomly inserted so that the total number of each pseudo spam message becomes some specific number (10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 are used in the experimentation). In Figure 8, hatched squares show the number of totally missed seed. Of all the seed inserted ones, inserted more than 40 times, were found. 25% of the seed inserted 10 times were found, and 75% were missed.

In the experiment, some occurrences of seed were missed and the recall rate (ratio of found and inserted spam) shows on a different line (white squares in Figure 8). When each seed is inserted 100 times, 100% of inserted spam are found. Since this number means that when a spammer sends 1000 commercial e-mail messages per day, all of them are detected. Since this seems to force a change in the current spammer's business model, we tentatively use 100% as the recall rate of our methods. Although the proposed method misses some spam when its occurrence is small (See Section4.3), both the precision and recall rate of our method are good enough in practical sense.

Figure 9 shows the cache consumption. The continuous line shows the hash database consumption (percentage) and the cross shows the direct-mapped cache consumption. After the direct-mapped cache is filled, the consumption of the hash database becomes stable and is about 90%.

When we designed these experimentations, we thought the similarity threshold (S in Table 2) as an important tuning parameter to handle randomized spam bodies. However, the preliminary experimentations show that "90%" works well, and we did not search the best value for this. Though this value seems to be good enough to handle today's spam through the mobile phone, the best value for other spam, such as spam though the internet, might be different. The parameter D, L, N, n, and S in Table 2 are the parameters which have to be tuned to handle different type of spam.

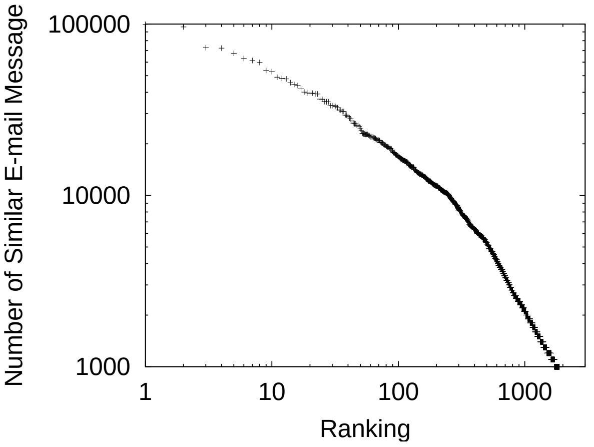


**Fig. 7** Distribution of Similar E-mail Message

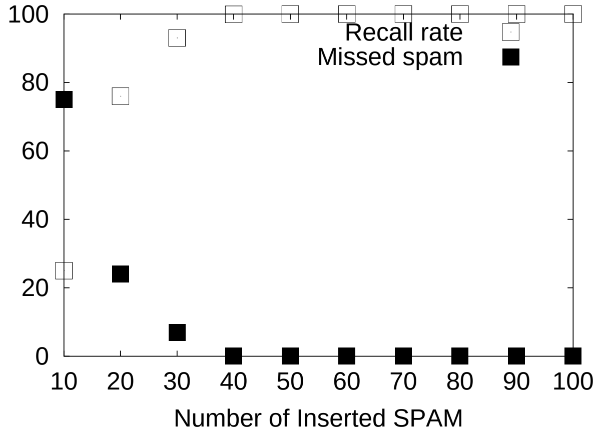


**Fig. 8** Recall Rate

## 4.2 Comparison with Conventional Methods

In the experiment shown in Table 2, our method could handle 13,361 mail messages per second (1.25 billion e-mail messages per day). None of the known spam filter seems to be able to handle over thousand e-mail messages per second. Additionally, most of them requires a supervisor for learning. Thus, as far as we know, our method is the only solution for our purpose, i.e, filtering out spam e-mail from regular mail traffic passing through our server without human maintenance.

Table 3 shows some comparisons. It shows the performance of the known methods on our mail data. First 10,000 mail messages are used for this experimen-

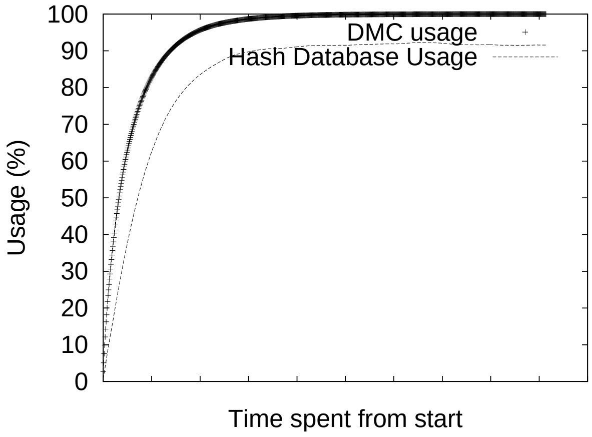


**Fig. 9** Cache Usage Log

| | SVM | NB | C4.5 | K-nn | bsfilter | SpamA. |
|---|---|---|---|---|---|---|
| Total CPU time (sec) | 756 | 79 | 254 | 10881 | 65 | 225 |
| Learning time (sec) | 744 | 22 | 244 | NA | 33 | NA |
| Memory Size (MB) | 191 | 70 | 58 | 81 | 327 | 192 |
| Recall (%) | 81 | 47 | 77 | 81 | 73 | 83 |
| Precision (%) | 99 | 97 | 95 | 100 | 98 | 22 |
| Speed Ratio | 1009 | 106 | 379 | 14528 | 87 | 300 |

**Table 3** Comparison with Other Methods

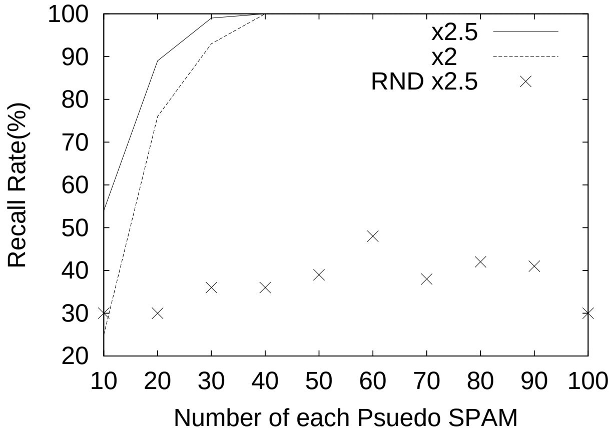


**Fig. 10** Effect of Memory Management

tation. To setup learning, spam messages, detected by our method, are marked as spam and others are marked as non-spam.

Support Vector Machine (SVM [14]), Naive Bayes (NB [15]), C4.5 [16], and K-Nearest Neighbor (K-nn [9]) are results of well-known machine learning methods. Weka [17] implementation are used in the experiments. Since these are general supervised learning methods, the contents of our hash database are used as the attributes. The first two thirds of the input are used for training. The latter one third of them are used for testing.

*bsfilter* [18] is an implementation of the method proposed in [6]. It is a slightly modified version so that it can handle Japanese. *SpamA* shows the results of *SpamAssassin* [7]. Although it does not require learning, the lack of Japanese handling results in its low precision. Since these two are spam filters, original e-mail messages are directly input into the system.

None of them have enough processing speed and none of them shows a recall rate of over 90%. Among them, SVM has the most desirable results (81% recall & 99% precision within a reasonable amount of CPU time). However, the difference between SVM's results and that of our method is significant. Also, the difference between supervised learning and unsupervised learning is significant, from a practical point of view. The most apparent difference is the CPU time required. SVM could only handle about 13.2 (= 10,000/756) e-mail messages per second. In other words, the required CPU time of SVM is 1009 times more than that of our method. Therefore SVM is too slow for our purposes.

### 4.3 Comparison of Memory Management Methods

Our method misses extremely rare spam messages. It also misses spam when used with a smaller sized hash database. Figure 10 and 11 show the recall rates of the proposed method on rare spam messages. They also show the improvement by the auxiliary memory management methods.

When the number of the direct-mapped cache entry (i.e., 2,000,000) is twice of the number of the hash database entry (i.e., 1,000,000), only the basic memory management controls the hash database entry, and all RND, RND2, LRU, LRU2 show the same performance (See Figure 10, dotted line). When the number of the direct-mapped cache entry is 2,500,000, the memory space for the hash database are fully used and the performance of RND2, LRU and LRU2 are slightly improved (solid line in Figure 10). However, the performance of RND decreased (dotted plots). The performance improvement of RND2, LRU and LRU2 is due to the fully used hash database. However, extra capacity of the direct-mapped cache (See Figure 12) disturbs the function of basic memory management and decreases the performance of RND.

Figure 11 shows the recall rates of the proposed method with a smaller size of hash database. Both LRU2 and RND2 works well with smaller size of hash database. In this experiment, each seed of pseudo spam is randomly inserted 40 times. Both LRU2 and RND2 works well with smaller size of hash database. Although LRU works well with a larger size of hash database, its recall rate decreases with 300,000 and less hash database entries. Both LRU2 and RND2 retain the entries which are observed more than once. This supports the advantage of the memory management strategy which retains the multiply accessed hash database entries.

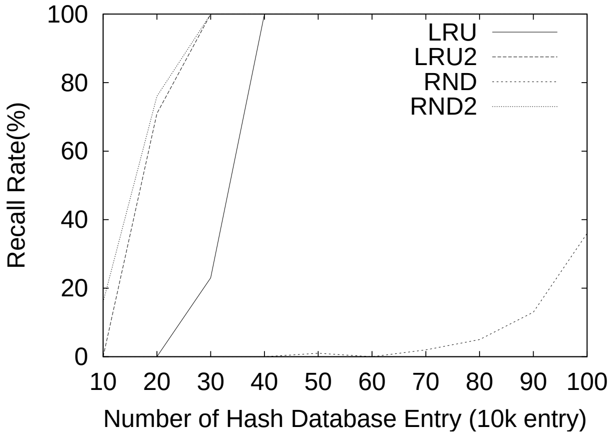


**Fig. 11** Effect of Hash Database Size

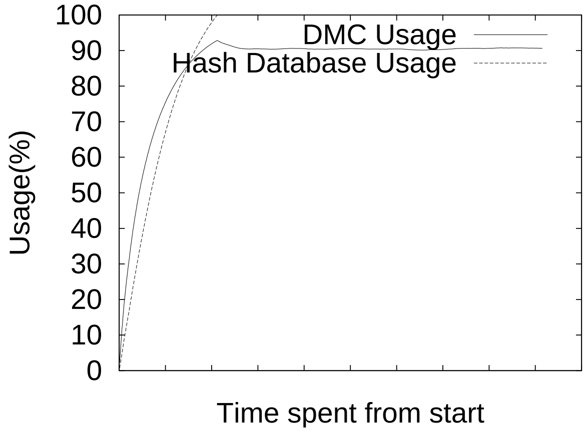


**Fig. 12** Cache Usage Log with extra DMC entry

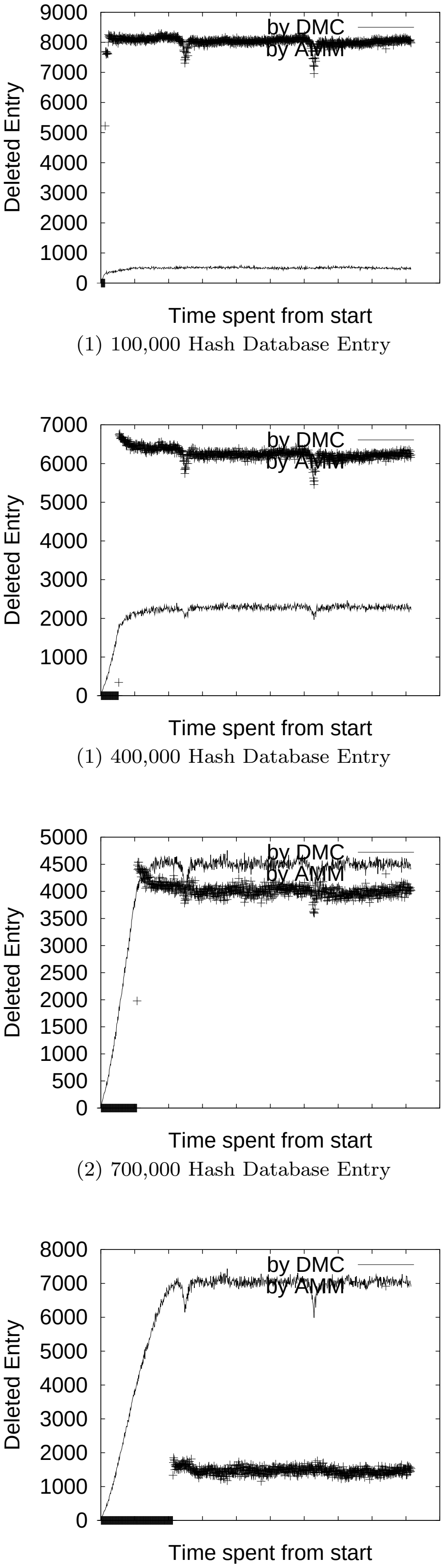
Figure 13 shows the number of deleted hash database entries during the experiments (RND2 with 2,500,000 DMC entry). Y-axis is the number and X-axis is the time spent from the experimentation start. Since the number of DMC entry was fixed to be 2,500,000, the auxiliary memory management methods have to delete much entries with fewer hash database entries.

Note that the performance shown in the previous subsection is good enough from practical viewpoints. The experimental results shown in this subsection are just to clarify the theoretical limitations and the possible improvements of the proposed method. As clearly shown, the performance of the proposed method can be further improved through appropriate memory management mechanisms.

## 5. Discussion

### 5.1 Accuracy & Evaluation Method

The results shown in Table 3 are worse than reported



(1) 100,000 Hash Database Entry

(1) 400,000 Hash Database Entry

(2) 700,000 Hash Database Entry

(3) 1,000,000 Hash Database Entry

**Fig. 13** Ratio of memory management

| | | SVM | NB | C4.5 | K-nn |
|---|---|---|---|---|---|
| 2/3 Train<br>1/3 Test | Recall (%) | 81 | 47 | 77 | 81 |
| | Precision (%) | 99 | 97 | 95 | 100 |
| 10 fold<br>CV | Recall (%) | 100 | 88 | 100 | 100 |
| | Precision (%) | 100 | 96 | 100 | 100 |
| Test =<br>Train | Recall (%) | 100 | 85 | 99 | 100 |
| | Precision (%) | 100 | 97 | 99 | 100 |

**Table 4** Testing methods

elsewhere [5]–[7]. Since most of them do not fully specify the experimental conditions and the data set used is different, we can not firmly conclude the reason for this difference. But experimental results shown in Table 4 show the difficulty in evaluating the performance of spam filters. Table 4 shows the same performance measures shown in Table 3. But it also shows the accuracy measured by 10 fold cross validation and accuracy on the training test.

Although cross validation is one of the standard ways to evaluate the accuracy of learning systems, it is not an appropriate way to evaluate the accuracy of spam filters. As shown in the Table, the accuracies measured by cross validation are similar to that on training data. Since a spammer sends a lot of the same spam message, random sampling of cross validation tends to make an equivalent training set and test set. Since most of the spam in the test set is contained in the training set, accuracy measured by the cross validation looks like the accuracy on the training set and is misleading.

However, in real situations, a spammer tends to create a new type of spam so that it can avoid spam filters. To emulate such situations, the first two thirds of data are used for training and the latter one third is used for testing. With sufficient amount of data it becomes a better method for evaluation. Table 3 shows such results.

To check what happens in a real life situation, we performed different experimentations. In the experimentations, we used e-mail messages in a public mailing list to emulate spam messages, and we used e-mail messages in a different mailing list to emulate ordinary messages. The use of a public mailing list enables us to undertake careful analysis of e-mail contents, which we are not able to perform on e-mail traffic discussed in the previous section.

The first mailing list has 528 messages (group S). The content of the messages relates to the Unix operating system. The second mailing list has 1583 messages (group H). The content of these messages relates to Chinese language. *bsfilter* was trained with all the messages in group H and half of the messages in group S. Half of the messages in group S was used to check the change of recall rate. Figure 14 shows the results.

As clearly shown in Figure 14, *bsfilter* missclassified most of the messages after some period, in addition the recall rate decreased. After careful analysis of the messages in group S, it is revealed that a new topic starts in that period. In the mailing list, the participants first discuss the general characteristics of Unix. Then, from that period onward, they start a new discussion about how to compile some specific program on Unix. This change in topic disturbs the analysis of *bsfilter* and reduces its recall rate.

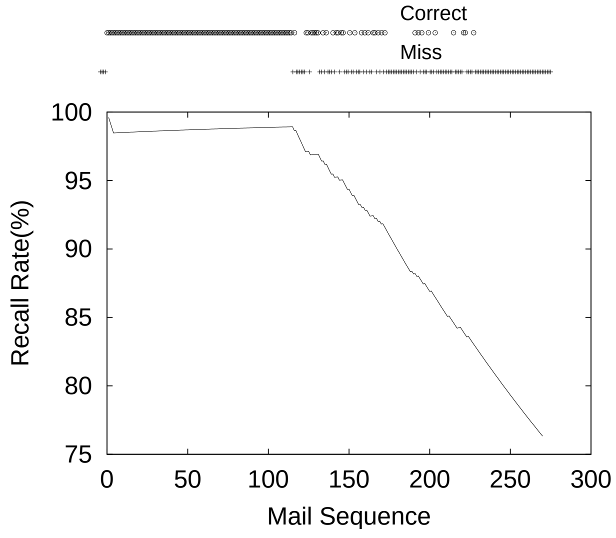


**Fig. 14** Effect of Topic Change

Although a new type of spam, which introduces a new product, seems to make a similar disturbance, it appears that cross validation cannot evaluate these phenomenon.

Note that the recall rate of *bsfilter* can be increased by on-line learning (See Figure 15). By reconstructing its database when it encounters new message, we can increase the recall rate of *bsfilter*. This emulates a more realistic situation of using a spam filter for personal use. However, cross validation cannot handle this situation.

### 5.2 Maintenance & Privacy

When we use supervised learning methods like [5] and [6], such methods require maintenance tasks. As shown in Figure 14, the accuracy of such methods decreases without maintenance. From the maintenance point of view, our unsupervised learning method has the following two advantages:

- Supervised learning methods require a positive and negative example of spam. This implies that someone has to check the contents of the mail manually and therefore has the potential to violate privacy. Since our method does not require any supervisors, user's privacy is inherently protected.
- Although our method requires a white list, maintaining such a white list is relatively easy, especially when comparing it to maintaining a black

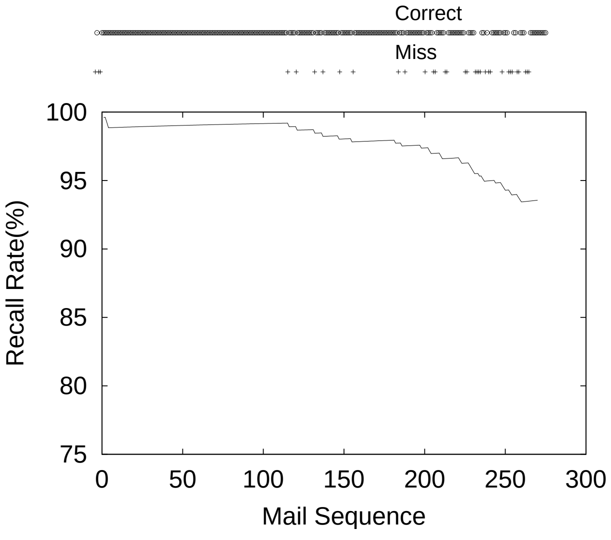


**Fig. 15** Effect of On-line Learning

list. The use of open mail relay servers and faking of the header information are common techniques of spammers. This makes the compilation of a black list difficult. None of the solicited mass e-mail messages are reported to fake its header information. The designing of a user interface which enables the user to declare solicited mass mail senders seems to be straight forward. Thus we choose a method which works with a white list.

Note that a white list can improve the precision by marking the mail messages from registered sites as non-spam messages. But it cannot improve the recall rate. To improve the recall rate, we need a black list. Since the recall rate of the conventional methods shown in Table 3 are relatively low, we need a black list if we wish to use the conventional methods. However, the required maintenance work of the black list prevent us from the use of conventional methods.

## 6. Conclusion

This paper reports on a new spam detection method which analyzes document space density to detect spam. The characteristics of this method are:

- High processing speed:
  With a single small desk-top computer, e.g., Pentium 4, 2.4 GHz with 825M bytes of memory, this method can handle over 13,000 e-mail messages per second (1.25 billion e-mail messages per day). Though known methods tend to require extensive computing resources, the performance of this proposed method is good enough to support a large mail server cluster of an ISP using a small PC. The direct-mapped cache method contributes to the efficient analysis of document space density.
- Maintenance free:
  Most of the traditional spam filtering methods requires maintenance of its database so that it can handle new types of spam. An unsupervised learning engine used in the proposed method can automatically update its database, and does not require such maintenance. Updating the database requires ISP operators to do a lot of maintenance work. Therefore, traditional spam filtering methods are not adequate enough to be used for e-mail servers that have a large number of customers.
- 100% recall rate and 100% precision:
  The results of our unsupervised learning engine might include both unsolicited and solicited mass e-mail messages. Since removing solicited mass e-mail message, e.g., a mail-based magazine, with a short white list, is not difficult, ISP operators are able to use this method as a perfect spam detector in a practical sense. Experimental results, which used over 50 million actual pieces of e-mail traffic prove this accuracy.
- Privacy protection:
  Hash based text representation and an unsupervised learning framework inherently protect user's privacy.

This paper also compares the basic memory management algorithm with possible refinements and shows the fact that the performance of the proposed method can be further improved through appropriate memory management mechanisms.

We believe that the proposed method is good enough to be used under practical situations. However, to fully utilize the proposed spam detection method to protect customers, the enrichment of anti spam and related law is necessary. The legal issues of mass e-mail are beyond the scope of our study.

### References

[1] G. Lindberg, RFC2505: Anti-Spam Recommendations for SMTP MTAs, IETF, 1999.
[2] G. Salton and M.J. McGill, Introduction to modern information retrieval, McGraw Hill, 1983.
[3] A.S. Tanenbaum, Structured Computer Organization (4th Edition), Prentice-Hall, 1999.
[4] Distributed checksum clearinghouse (http://www.rhyolite.com/anti-spam/dcc/), 2003.
[5] P. Graham, "Better bayesian filtering," Proc. of the 2003 Spam Conference, 2003.
[6] G. Robinson, "Spam detection (http://radio.weblogs .com/0101454/stories/2002/09/16/spamdetection.html)," 2002.
[7] "SpamAssassin (http://useast.spamassassin.org/)," 2003.
[8] T. Wada, S. Saito, Y. Izumi, and T. Uehara, "Contents based Mass-Mail Filtering (In Japanese)," IPSJ SIG Technical Report, pp.55–60, 2003.
[9] T. Mitchell, Machine Learning, McGraw Hill, 1997.
[10] D.E. Knuth, The Art of Computer Programming, Vol.3 – Sorting and Searching, Addison-Wesley, 1973.
[11] F. Adachi, T. Washio, H. Motoda, and H. Hanafusa, "Development and Application of Generic Search method Based

on Transformation Invariance (In Japanese)," Proc. of the 10th Annual Conference of Japanese Society for Artificial Intelligence, pp.1E3–04, 2003.

[12] T. Raita and J. Teuhola, "Predictive text compression by hashing," Proc. of ACM Conference on Information Retrieval, New Orleans, 1987.

[13] N. Nishikawa, T. Hosakowa, Y. Mori, K. Yoshida, and H. Tsuji, "Memory-based architecture for distributed www caching proxy," Proc. of World Wide Web Conference 98, pp.205–214, 1998.

[14] J. Platt, Fast training of support vector machines using sequential minimal optimization, pp.185–208, MIT Press, 1999.

[15] G.H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp.338–345, Morgan Kaufmann Publishers, San Mateo, 1995.

[16] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[17] Weka 3: Machine learning software in java (http://www.cs.waikato.ac.nz/ml/weka/), 2003.

[18] bsfilter, "(http://www.h2.dion.ne.jp/nabeken/bsfilter/)," 2003.

**Kenichi YOSHIDA** m

**Fuminori ADACHI** n

**Takashi WASHIO** n

**Hiroshi MOTODA** n

**Teruaki HOMMA** n

**Akihiro NAKASHIMA** n

**Hiromitsu FUJIKAWA** m

**Katsuyuki YAMAZAKI** m