Density-Based Spam Detector

Kenichi YOSHIDA
Graduate School of Business Science, University of Tsukuba
Otsuka 3-29-1, Bunkyo, Tokyo 112-0012, Japan
yoshida@gssm.otsuka.tsukuba.ac.jp

Fuminori ADACHI
Takashi WASHIO
Hiroshi MOTODA
ISIR, Osaka University
8-1, Mihogaoka, Ibarakishi, Osaka 567-0047, Japan
{adachi,washio,motoda}@ar.sanken.osaka-u.ac.jp

Teruaki HOMMA
Akihiro NAKASHIMA
KDDI Corporation
Garden Air Tower, 3-10-10, Iidabashi, Chiyoda, Tokyo 102-8460, Japan
{teruaki,nakasima}@kddi.com

Hiromitsu FUJIKAWA
Katsuyuki YAMAZAKI
KDDI R&D Laboratories Inc.
2-1-15 Ohara, Kami-fukuoka, Saitama 356-8502, Japan
{fujikawa,yamazaki}@kddilabs.jp

ABSTRACT

The volume of mass unsolicited electronic mail, often known as spam, has recently increased enormously and has become a serious threat to not only the Internet but also to society. This paper proposes a new spam detection method which uses document space density information. Although it requires extensive e-mail traffic to acquire the necessary information, an unsupervised learning engine with a short white list can achieve a 98% recall rate and 100% precision. A direct-mapped cache method contributes handling of over 13,000 e-mails per second. Experimental results, which were conducted using over 50 million actual e-mails of traffic, are also reported in this paper.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Data Mining; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Performance, Experimentation, Security

Keywords

spam, unsupervised learning, document space density, direct-mapped cache

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

1. INTRODUCTION

Mass unsolicited electronic mail, often known as spam [7][1], has recently increased enormously and has become a serious threat to not only the Internet but also to society. This is especially true in Japan, where mobile phones have e-mail capability and e-mails 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-mails per day. This means that the spam filter has to handle more than 1000 e-mails per second. Since the most well known spam filter program requires 10 to 100 milliseconds to deal with each e-mail, performance improvement is necessary.

- Ease of maintenance:
  Most of the traditional spam filtering methods requires maintenance of their data-base so that they can handle new types of spam. Unfortunately, spammers tend

[1]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 data base 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 mails. See Section 5.2 for details.

to be very productive and are always producing new type 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 spam filter for clients, 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 for clients and servers. A server requires miss-judgment probability of normal e-mail 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 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 [15] information, and 2) an efficient implementation of the first idea through the use of a direct-mapped cache [17]. 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. Section 4 reports on the experimental results that used over 50 million actual pieces of e-mail traffic. It also compares our method with other methods. Section 5 discusses related topics. Finally, Section 6 concludes our findings.

2. RELATED WORKS

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., [3, 4, 14, 16], and [18]. Some of them use a Bayesian-like approach [4, 14], or a rule-based approach [16], and some use a checksum data base [3, 18] to detect spam.

Vector representation, e.g., TF IDF[15], combined with machine learning techniques [8] 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-mails. 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 e-mail with small 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 mail on this representation alone is not a simple task.

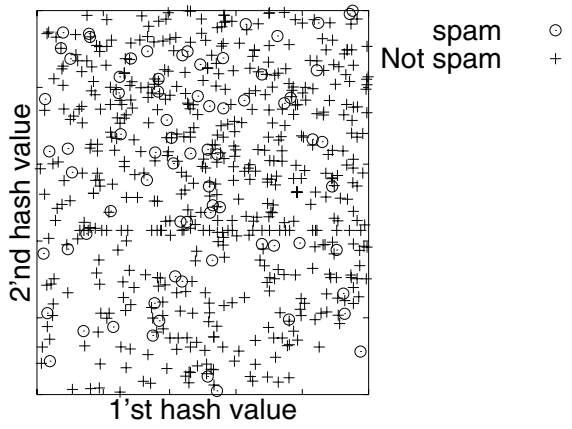

Figure 1: Vector Representation of E-mail

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 [6] and it is used for a variety of purposes, e.g., text retrieval [1], text compression [13], and spam filtering[3, 18]. 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 [17] is the core of the engine and it is used as a substitute for the LRU cache. LRU performance on heavily maldistributed data is studied in various network applications. For example, [10] analyzes WWW traffic and reveals LRU's high performance on gathering maldistributed WWW data. In our study, the direct-mapped cache [17] is used to gather maldistributed spam mails.

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.

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 [15] as the key piece of information to distinguish spam from other e-mails. More precisely, we just count the number of similar e-mails. By counting the number of similar mails, we can estimate the local document density around the mail.

Figure 2 shows the histogram of e-mails shown in Figure 1. The X and Y axis are that of Figure 1. The Z axis

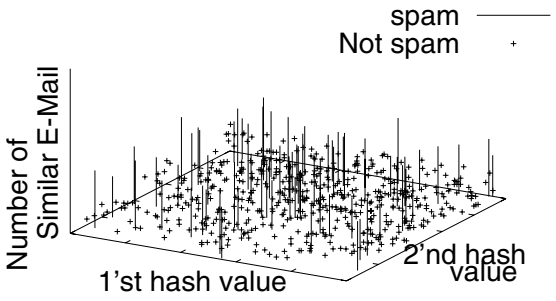

Figure 2: Histogram of Similar Mails

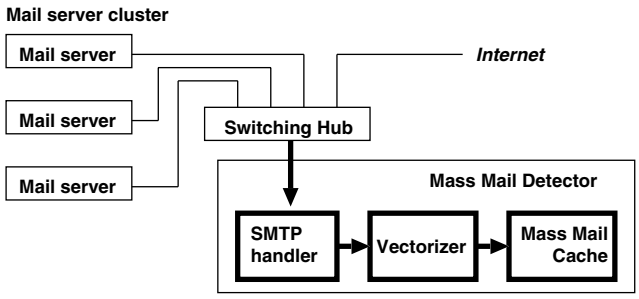

Figure 3: Configuration of Mass-Mail Detector

is the number of similar e-mails. As clearly shown in the Figure, the use of the histogram makes distinguishing spam from other e-mails far easier. Actually, experimental results reported in Section 4 showed that simple threshold is enough to distinguish spam from other e-mails.

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-mails, spammers have to send the same spam far more than that. Note that some of the unethical spam mail are said to be difficult to judge even for a human. However, the existence of over thousand identical e-mails 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-mails. Then, Vectorizer transfers the text into vector representation.

There are various vector representations, such as term frequency and N gram [9]. Although they are candidate representation, we use a hash-based vector representation (See Figure 4). From each e-mail, hash values of each length L

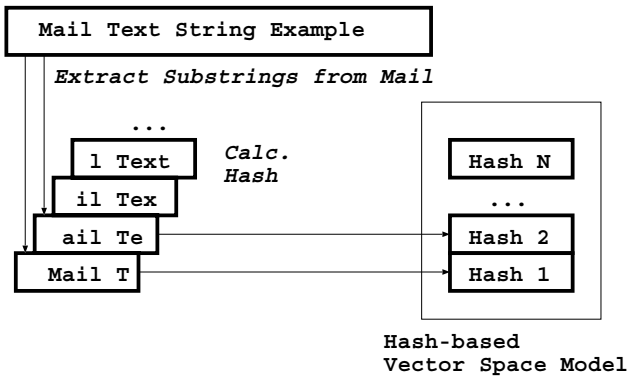

Figure 4: Hash-based Vector Representation

substring are calculated,[2] and then the first N of them are used as vector representation of the e-mail.

Japanese, English and other languages are used in mobile phone e-mails in Japan. Bigram [9] 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-mails per second and it needs to check a million previous e-mails to handle the current single e-mail, 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 [17] architecture to speed up processing.

Figure 5 shows the data structure and Figure 6 shows the algorithm. The hash data base in Figure 5 stores the hash values of each e-mail and the number of similar e-mails. The direct-mapped cache copies the n% of the hash values. It also stores the pointer to the e-mail's entry in the hash data base. To check a single piece of e-mail, in order to find similar previous e-mail 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-mails which have the same hash value through this cache.

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-mails, the algorithm deletes the entry in the hash data base for the overwritten e-mail so that it can reuse the memory space of the hash data base.

Although this architecture does not have an explicit LRU cache mechanism to control entries in the hash data base, the overwrite mechanism of direct-mapped cache controls the entry in the hash data base as if it were controlled by a LRU mechanism.

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

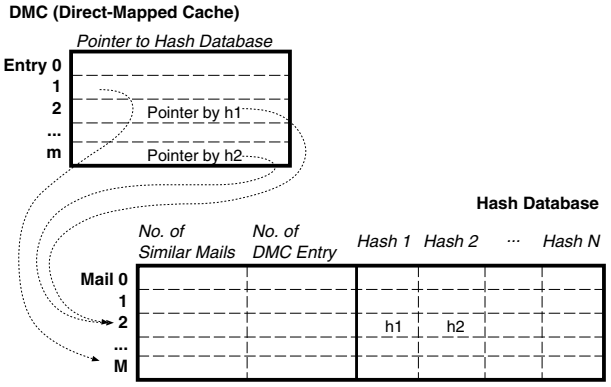

Figure 5: Data Structure of Mass Mail Cache

Total Number of e-mail	53,985,002
Total Number of "spam"	12,324,762
DMC entry	2,000,000
Hash Data Base entry	1,000,000
"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)	98 %
Precision	100 %

Table 1: Summary of Experimental Results

4. EXPERIMENTAL RESULTS

Unfortunately, we are not allowed to disclose all of our findings 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 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 1 summarizes the results. 22.8% of the total number of e-mails are spam, and the distribution of similar e-mail is shown in Figure 7. As seen in various network data, [10] for example, the distribution of similar e-mail follows Zipf's law (See Figure 7).

The similarity threshold used in the experimentations is 90%. E-mails transferred more than 100 times are marked as spam. 100% of marked mails are mass mails as is defined. One million entries for the hash data base 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. 10% of the hash values in the hash data base are copied in the direct-mapped cache.

```
Main Procedure Check-Mail
Input
    T: Text of Mail
Var h: Hash value
begin
    New-Hash-DB-Candidate
        ← Make N Hash values from T
    for h in New-Hash-DB-Candidate do
        if Similar(Mail in Hash-DB pointed by h,
                New-Hash-DB-Candidate)
        then Update-Similar-Mail(
                Mail in Hash-DB pointed by h)
            exit Check-Mail
    // If No Similar Entry exists in Hash DB
    Store-New-Mail(New-Hash-DB-Candidate)
end

Function Similar
Input
    H1: Hash-DB 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-DB entry
Var h: Hash value
begin
    Increment "No.of Similar Mail" of H1
    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 DB Entry
    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-DB Entry
begin
    Set DMC Entry for h so that it points H
    if Previous h already point Hash-DB Entry: e
        & e ≠ H
    then Decrement "No.of DMC Entry" of e
    if "No.of DMC Entry" of e = 0
    then Delete e from Hash DB
        & Clear DMC entry which point e
end
```

Figure 6: Mass Mail Caching Algorithm

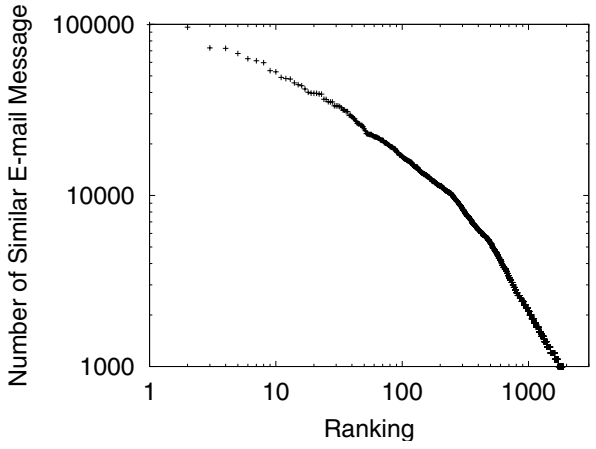

Figure 7: Distribution of Similar E-mail

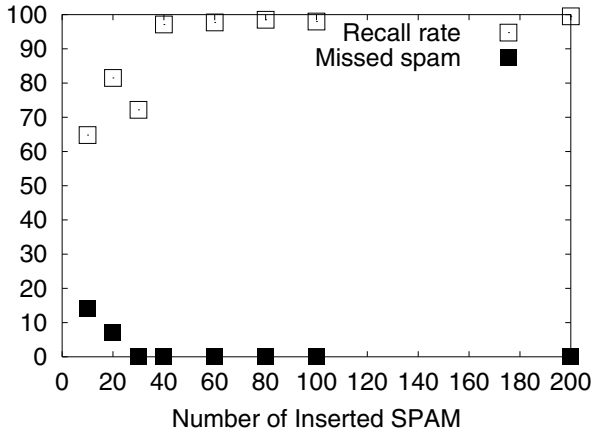

Figure 8: Recall Rate

Note that the results might include both unsolicited and solicited mass e-mails. However removing solicited mass e-mail, 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 mails (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-mails are extracted and merged with pseudo spam e-mail. The preparatory experimentation measures the recall rate over this pseudo spam e-mail (See Figure 8).

100 seeds of spam are prepared and each seed is randomly inserted so that the total number of each pseudo spam becomes some specific number (10, 20, 30, 40, 60, 80, 100 and 200 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 30 times were found. Half of the seed inserted 10 times were found, but half were missed.

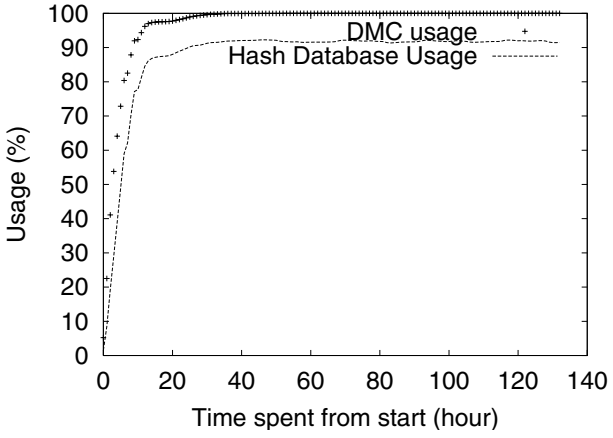

Figure 9: Cache Usage Log

	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 2: Comparison with Other Methods

In the experiment, some occurrences of seed are missed and the recall rate (ratio of found and inserted spam) shows a different line (white squares in Figure 8). When each seed is inserted 100 times, 98% of inserted spam e-mail are found. Since this number means that when a spammer sends 1000 commercial e-mails per day, 98% of them are detected. Since this seems to force a change in the current spammer's business model, we tentatively use 98% as the recall rate of our methods. In practical sense, both the precision and recall rate of our method are good enough.

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

4.2 Performance Comparison

In the experiment shown in Table 1, our method could handle 13,361 mails per second (1.25 billion e-mails per day). None of the known spam filter seems to be able to handle over thousand mails 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 2 shows some comparisons. It shows the performance of the known method on our mail data. First 10,000 mails are used for this experimentation. To setup learning, spam e-mails, detected by our method, are marked as spam and others are marked as non-spam.

Support Vector Machine (SVM [11]), Naive Bayes (NB [5]), C4.5 [12], and K-Nearest Neighbor (K-nn [8]) are re-

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

Table 3: Testing methods

sults of well-known machine learning methods. Weka [19] implementation are used in the experiments. Since these are general supervised learning methods, the contents of our hash data base 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 [2] is an implementation of the method proposed in [14]. It is a slightly modified version so that it can handle Japanese. *SpamA* shows the results of *SpamAssassin* [16]. Although it does not require learning, the lack of Japanese handling results in its low precision. Since these two are spam filters, original e-mails 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-mails 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.

5. DISCUSSION

5.1 Accuracy & Evaluation Method

The results shown in Table 2 are worse than reported elsewhere [4, 14, 16]. 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 3 show the difficulty in evaluating the performance of spam filters. Table 3 shows the same performance measures shown in Table 2. 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, 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 this sufficient amount of data it becomes a better method for evaluation. Table 2 shows such results.

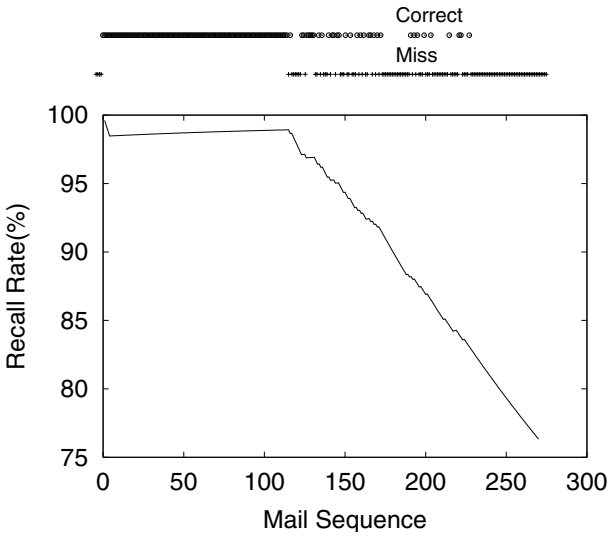

Figure 10: Effect of Topic Change

To check what happens in a real life situation, we performed different experimentations. In the experimentations, we used e-mails in a public mailing list to emulate spam e-mails, and we used e-mails in a different mailing list to emulate ordinary e-mails. 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 e-mails (group S). The content of the e-mails relates to the Unix operating system. The second mailing list has 1583 e-mails (group H). The content of these e-mails relates to Chinese language. *bsfilter* was trained with all the e-mails in group H and half of the e-mails in group S. Half of the e-mails in group S was used to check the change of recall rate. Figure 10 shows the results.

As clearly shown in Figure 10, *bsfilter* miss-classified most of the e-mails after some period, in addition the recall rate decreased. After careful analysis of the e-mails 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.

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 11). By reconstructing its data base when it encounters new e-mail, 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.

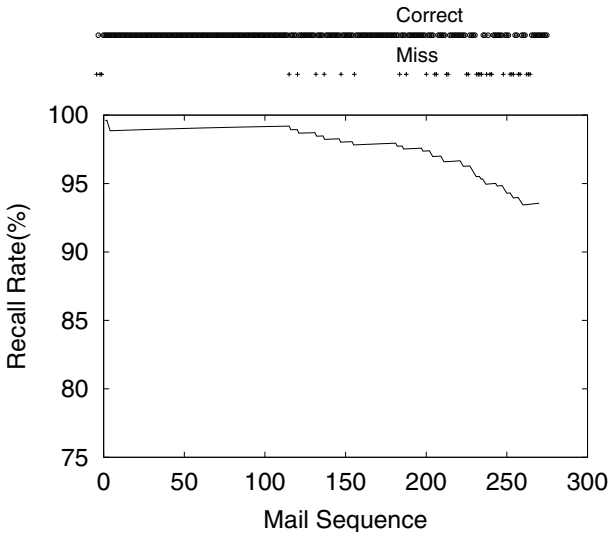

Figure 11: Effect of On-line Learning

5.2 Maintenance & Privacy

When we use supervised learning methods like [4, 14], such methods require maintenance tasks. As shown in Figure 10, 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 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-mails 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.

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-mails per second (1.25 billion e-mails per day). Though known methods tend to require too much 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 data-base so that it can handle new types of spam. An unsupervised learning engine used in the proposed method can automatically update its data-base, and does not require such maintenance. Updating the data-base requires a lot of maintenance work for ISP operators. Therefore, traditional spam filtering methods are not adequate enough to be used for e-mail servers that have a large number of customers.
- 98% recall rate and 100% precision:
  The results of our unsupervised learning engine might include both unsolicited and solicited mass e-mails. Since removing solicited mass e-mail, e.g., a mail-based magazine, with a short white list, is not difficult, ISP operators are able to use this method as perfect 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.

At the time of writing this paper, we finished the research phase and started to develop a system for daily services to protect users. 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.

7. REFERENCES

[1] F. Adachi, T. Washio, H. Motoda, and H. Hanafusa. Development and Application of Generic Search method Based on Transformation Invariance (In Japanese). In *Proc. of the 10th Annual Conference of Japanese Society for Artificial Intelligence*, pages 1E3–04, 2003.

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

[3] Distributed checksum clearinghouse (http://www.rhyolite.com/anti-spam/dcc/), 2003.

[4] P. Graham. Better bayesian filtering. In *Proc. of the 2003 Spam Conference*, 2003.

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

[6] D. E. Knuth. *The Art of Computer Programming, Vol.3 – Sorting and Searching*. Addison-Wesley, 1973.

[7] G. Lindberg. *RFC2505: Anti-Spam Recommendations for SMTP MTAs*. IETF, 1999.

[8] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[9] R. Mitkov, editor. *The Oxford Handbook of Computational Linguistics*. Oxford Press, March 2003.

[10] N. Nishikawa, T. Hosakowa, Y. Mori, K. Yoshida, and H. Tsuji. Memory-based architecture for distributed

www caching proxy. In *Proc. of World Wide Web Conference 98*, pages 205–214, 1998.

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

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

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

[14] G. Robinson. Spam detection (http://radio.weblogs .com/0101454/stories/2002/09/16/spamdetection.html), 2002.

[15] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw Hill, 1983.

[16] SpamAssassin (http://useast.spamassassin.org/), 2003.

[17] A. S. Tanenbaum. *Structured Computer Organization (4th Edition)*. Prentice-Hall, 1999.

[18] T. Wada, S. Saito, Y. Izumi, and T. Uehara. Contents based Mass-Mail Filtering (In Japanese). In *IPSJ SIG Technical Report*, pages 55–60, 2003.

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