IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, p-ISSN: 2278-8719, www.iosrjen.org
Volume 2, Issue 9 (September 2012), PP 55-60

Cosdes: A Collaborative Spam Detection System with a Novel E-Mail Abstraction Scheme

G.Surekha[1], Y.Sowjanya Kumari[2] , Dr.P.Harini[3]
[1]II M.Tech C.S.E , St.Anns College Of Engineering &Technology, Chirala.
[2]M.Tech (C.S.E), Associate Professor, St.Anns College Of Engineering &Technology, Chirala.
[3]Ph.D, Professor , St.Anns College Of Engineering &Technology, Chirala.

Abstract—*E-mail communication is indispensable nowadays, but the e-mail spam problem continues growing drastically. In recent years, the notion of collaborative spam filtering with near-duplicate similarity matching scheme has been widely discussed. The primary idea of the similarity matching scheme for spam detection is to maintain a known spam database, formed by user feedback, to block subsequent near-duplicate spams. On purpose of achieving efficient similarity matching and reducing storage utilization, prior works mainly represent each e-mail by a succinct abstraction derived from e-mail content text. However, these abstractions of e-mails cannot fully catch the evolving nature of spams, and are thus not effective enough in near-duplicate detection. In this paper, we propose a novel e-mail abstraction scheme, which considers e-mail layout structure to represent e-mails. We present a procedure to generate the e-mail abstraction using HTML content in e-mail, and this newly devised abstraction can more effectively capture the near-duplicate phenomenon of spams. Moreover, we design a complete spam detection system Cosdes (standing for COllaborative Spam Detection System), which possesses an efficient near-duplicate matching scheme and a progressive update scheme. The progressive update scheme enables system Cosdes to keep the most up-to-date information for near-duplicate detection. We evaluate Cosdes on a live data set collected from a real e-mail server and show that our system outperforms the prior approaches in detection results and is applicable to the real world.*

I. INTRODUCTION

EMAIL communication is prevalent and indispensable nowadays. However, the threat of unsolicited junk emails, also known as spams, becomes more and more serious. According to a survey by the website Top Ten REVIEWS [11], 40 percent of e-mails were considered as spams in 2006. The statistics collected by MessageLabs1 show that recently the spam rate is over 70 percent and persistently remains high. The primary challenge of spam detection problem lies in the fact that spammers will always find new ways to attack spam filters owing to the economic benefits of sending spams. Note that existing filters generally perform well when dealing with clumsy spams, which have duplicate content with suspicious keywords or are sent from an identical notorious server. Therefore, the next stage of spam detection research should focus on coping with cunning spams which evolve naturally and continuously.

Definition 1 (<mytext=>). <mytext=> is a newly defined tag that represents a paragraph of text without any HTML tag embedded.

Since we ignore the semantics of the text, the proposed abstraction scheme is inherently applicable to e-mails in all languages. This significant feature is superior to most existing methods. Once e-mails are represented by our newly devised e-mail abstractions, two e-mails are viewed as near-duplicate if their HTML tag sequences are exactly identical to each other. Note that even when spammers insert random tags into e-mails, the proposed e-mail abstraction scheme will still retain efficacy since arbitrary tag insertion is prone to syntax errors or tag mismatching, meaning that the appearance of the e-mail content will be greatly altered. Moreover, the proposed procedure SAG also adopts some heuristics to better guarantee the robustness of our approach. While a more sophisticated e-mail abstraction is introduced, one challenging issue arises: how to efficiently match each incoming e-mail with an existing huge spam database. To resolve this issue, we devise an innovative tree structure, SpTrees, to store large amounts of the e-mail abstractions of reported spams, and SpTrees contribute to substantially promoting the efficiency of matching. In the design of the near-duplicate matching scheme based on SpTrees, we aim at reducing the number of spams and tags which are required to be compared.

II. PRELIMINARIES

In this section, the definition of near-duplicate, in this paper, is presented in Section 2.1. We then review the related works on spam detection in Section 2.2.

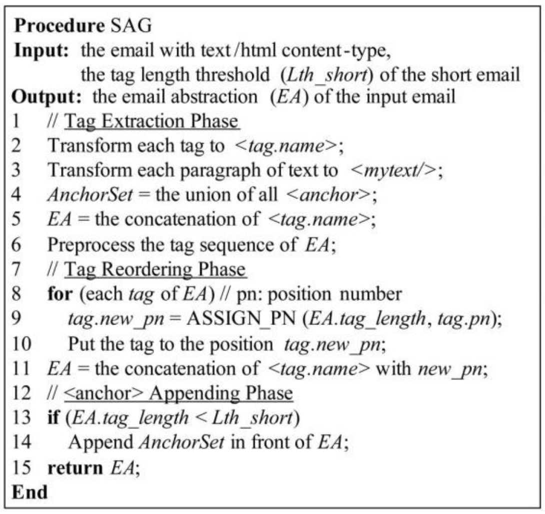
2.1 Definition of Near-Duplicate

The central idea of near-duplicate spam detection is to exploit reported known spams to block subsequent ones which havesimilar content. For different forms of e-mail representation, the definitions of similarity between two e-mails are diverse. Unlike most prior works representing e-mails based mainly on content text, we investigate representing each e-mail using an HTML tag sequence, which depicts the layout structure of e-mail, and look forward to more effectively capturing the near-duplicate phenomenon of spams. Initially, the definition of <anchor> tag is given as follows:

Definition 2 (<anchor>). The tag <anchor> is one type of newly defined tag that records the domain name or the e-mail address in an anchor tag. For example, the anchor tag <a href="http://arbor.ee. ntu.edu.tw/index.htm"> is transformed to <arbor.ee.ntu. edu.tw>. The anchor tag <a href="mailto:cytseng@arbor. ee.ntu.edu.tw"> is transformed to <cytseng@arbor.ee. ntu.edu.tw>. The purpose of creating the <anchor> tag is to minimize the false positive rate when the number of tagsin an e-mail abstraction is short. The less the number of tags in an e-mail abstraction, the more possible that a ham may be matched with known spams and be misclassified as a spam. Therefore, when the number of tags in an e-mail abstraction is smaller than a predefined threshold, for each anchor tag <a>, we specifically record the targeted domain name or e-mail address, which is a significant clue for identifying spams.

1.2 Related Works

Since the e-mail spam problem is increasingly serious nowadays, various techniques have been explored to relieve the problem. Based on what features of e-mails are being used, previous works on spam detection can be generally classified into three categories: 1) content-based methods, 2) noncontent-based methods, and 3) others. Initially, researchers analyze e-mail content text and model this problem as a binary text classification task. Representatives of this category are Naive Bayes [14], [20] and Support Vector Machines (SVMs) [1], [10], [15], [27] methods. In general, Naive Bayes methods train a probability model using classified e-mails, and each word in e-mails will be given a probability of being a suspicious spam keyword. As for SVMs, it is a supervised learning method, which possesses outstanding performance on text classification tasks. Traditional SVMs [10] and improved SVMs [1], [15], [27] have been investigated. While above conventional machine learning techniques have reported excellent results with static data sets, one major disadvantage is that it is cost-prohibitive for large-scale applications to constantly retrain these methods with the latest information to adapt to the rapid evolving nature of spams. The spam detection of these methods on the e-mail corpus with various language has been less studied yet. In addition, other classification techniques, including markov random field model [3], neural network [6] and logic regression [2], and certain specific features, such as URLs [26] and images [19], [29] have also been taken into account for spam detection.



```
Procedure SAG
Input:  the email with text/html content-type,
        the tag length threshold (Lth_short) of the short email
Output: the email abstraction (EA) of the input email
1   // Tag Extraction Phase
2   Transform each tag to <tag.name>;
3   Transform each paragraph of text to <mytext/>;
4   AnchorSet = the union of all <anchor>;
5   EA = the concatenation of <tag.name>;
6   Preprocess the tag sequence of EA;
7   // Tag Reordering Phase
8   for (each tag of EA) // pn: position number
9       tag.new_pn = ASSIGN_PN (EA.tag_length, tag.pn);
10      Put the tag to the position tag.new_pn;
11  EA = the concatenation of <tag.name> with new_pn;
12  // <anchor> Appending Phase
13  if (EA.tag_length < Lth_short)
14      Append AnchorSet in front of EA;
15  return EA;
End
```

Fig. 1. Algorithmic form of procedure SAG.

The authors make use of spam-vocabulary patterns produced by Teiresias pattern discovery algorithm. In [16], the I-Match signature determined by a set of unique terms shared by spams and the I-Match lexicon is put to use. In [21], the content similarity of e-mails computed using extracted words is measured. It is noted that most existing methods generate e-mail abstractions based mainly on content text. However, randomized and normal paragraphs are commonly inserted in spams nowadays, and thus if an e-mail abstraction is generated by the whole content text, the near-duplicate part of spams cannot be captured. Moreover, generating e-mail abstraction with the content text also suffers from the problem of not being applicable to all languages.

III. E-MAIL ABSTRACTION SCHEME

In this section, a novel e-mail abstraction scheme is introduced. In Section 3.1, procedure SAG is presented to depict the generation process of an e-mail abstraction. The devised data structures SpTable and SpTrees are illustrated in Section 3.2. Finally, the robustness issue is discussed in Section 3.3.

3.1 Structure Abstraction Generation

Wepropose the specific procedureSAGto generate the e-mail abstraction using HTML content in e-mail. SAG is elaborated with the example of Fig. 3, and the algorithmic form of SAG is outlined in Fig. 1. Procedure SAG is composed of three major phases, Tag Extraction Phase, Tag Reordering Phase, and <anchor> Appending Phase. In Tag Extraction Phase, the name of each HTML tag is extracted, and tag attributes and attribute values are eliminated. In addition, each paragraph of text without any tag embedded is transformed to <mytext=>. In lines 4-5, <anchor> tags are then inserted into AnchorSet, and the first 1,023 valid tags are concatenated

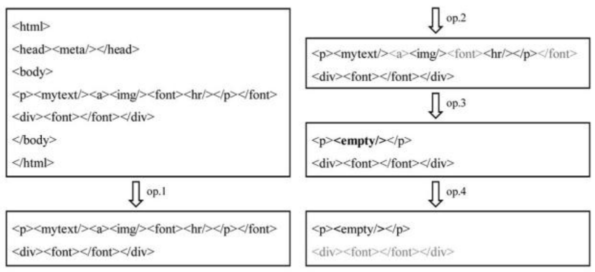


Fig. 2. An example of the preprocessing step in Tag Extraction Phase of procedure SAG.

To form the tentative e-mail abstraction. Note that we retain only the first 1,023 tags as the tag sequence. The main reason is that the rear part of long e-mails can be ignored without affecting the effectiveness of near-duplicate matching. Subsequently, in line 6 of Fig. 1,wepreprocess the tag sequence of the tentative e-mail abstraction. One objective of this preprocessing step is to remove tags that are common but not discriminative between e-mails. The other objective is to prevent malicious tag insertion attack, and thus the robustness of the proposed abstraction scheme can be further enhanced.

3.2 Design of SpTable and SpTrees

One major focus of this work is to design the innovative data structure to facilitate the process of near-duplicate matching. SpTable and SpTrees (sp stands for spam) are proposed to store large amounts of the e-mail abstractions of reported spams. As shown in Fig. 4, several SpTrees are the kernel of the database, and the e-mail abstractions of collected spams are maintained in the corresponding SpTrees. According to Definition 3, two e-mail abstractions are possible to be near-duplicate only when the numbers of their tags are identical. Thus, if we distribute e-mail abstractions with different tag lengths into diverse SpTrees, the quantity of spams required to be matched will decrease. However, if each SpTree is only mapped to one single tag length, it is too much of a burden for a server to maintain such thousands of SpTrees. In view of this concern, each SpTree is designed to take charge of e-mail abstractions within a range of tag lengths. As can be seen in Fig. 4, SpTable is created to record overall information of SpTrees. The ith column of SpTable links to the root of SpTree_i by a pointer, and e-mail abstractions with tag lengths ranging from 2i to 2iþ1 _ 1 belong to SpTree_i.

3.3 Robustness Issue

The main difficulty of near-duplicate spam detection is to withstand malicious attack by spammers. Prior approaches generate e-mail abstractions based mainly on hash-based content text. These methods primarily differ in what granularity is used as the input of the hash function. For example, the authors in [16], [21], [22] extract words or terms to generate the e-mail abstraction. Besides, substrings extracted by various techniques are widely employed in [7], [8], [12], [17], [23], [25], [30], [31]. However, this type of email representation inherently has following disadvantages. First, the insertion of a randomized and normal paragraph can easily defeat this type of spam filters. Moreover, since the structures and features of different languages are diverse, word and substring extraction may not be applicable to e-mails in all languages. Concretely speaking, for instance, trigrams of substrings used in [7], [8], [17] are not suitable for nonalphabetic languages, such as

Chinese. In this paper, we devise a novel e-mail abstraction scheme that considers e-mail layout structure to represent e-mails. To assess the robustness of the proposed scheme, we model possible spammer attacks and organize these attacks as following three categories. Examples and the outputs of preprocessing of procedure SAG are shown in Fig. 6.

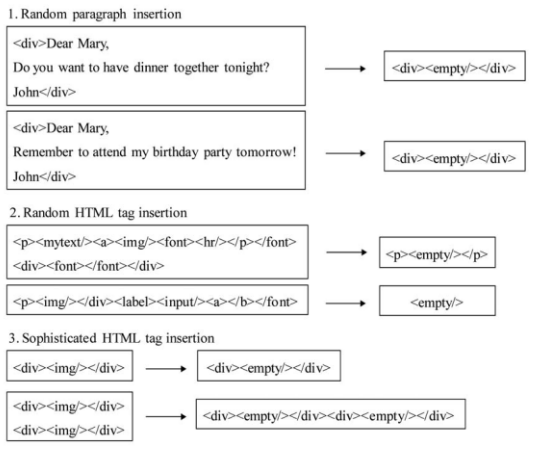

1. Random paragraph insertion
<div>Dear Mary,
Do you want to have dinner together tonight?
John</div>
<div><empty/></div>
<div>Dear Mary,
Remember to attend my birthday party tomorrow!
John</div>
<div><empty/></div>
2. Random HTML tag insertion
<p><mytext/><a><img/><font><hr/></p></font>
<div><font></font></div>
<p><empty/></p>
<p><img/></div><label><input/><a></b></font>
<empty/>
3. Sophisticated HTML tag insertion
<div><img/></div>
<div><empty/></div>
<div><img/></div>
<div><img/></div>
<div><empty/></div><div><empty/></div>

Fig. 6. Examples of possible spammer attacks.

Advertisement keywords are inserted to confuse textbased spam filtering techniques. It is noted that our scheme transforms each paragraph into a newly created tag <mytext=>, and consecutive empty tags will then be transformed to <empty=>. As such, the representation of each random inserted paragraph is identical, and thus our scheme is resistant to this type of attack.

3.3.1 Random HTML Tag Insertion

If spammers know that the proposed scheme is based on HTML tag sequences, random HTML tags will be inserted rather than random paragraphs. On the one hand, arbitrary tag insertion will cause syntax errors due to tag mismatching. This may lead to abnormal display of spam content that spammers do not wish this to happen. On the other hand, procedure SAG also adopts some heuristics (as depicted in Section 3.1) to deal with the random insertion of empty tags and the tag mismatching of nonempty tags. Fig. 6 shows two example outputs and the details of each step can be found in Fig. 2. With the proposed method, most random inserted tags will be removed, and thus the effectiveness of the attack of random tag insertion is limited. We shall verify this inference in Section 5.4.

3.3.2 Sophisticated HTML Tag Insertion

Suppose that spammers are more sophisticated, they may insert legal HTML tag patterns. As hown in Fig. 6, if tag patterns that do conform to syntax rules are inserted, they will not be eliminated. However, although some crafty tricks may be conceivable, it is not intuitive for spammers to generate a large number of spams with completely distinct e-mail layout structure. Note that due to space limitation, we are not able to discuss all possible situations. Nevertheless, representing emails with layout structure is more robust to most existing attacks than text-based approaches. Even though new attack has been designed, we can react against it by adjusting the preprocessing step of procedure SAG. On the other hand, our approach extracts only HTML tag sequences and transforms each paragraph with no tag embedded to <mytext=>, meaning that the proposed abstraction scheme can be applied to e-mails in all languages without modifying any components. This important feature also enables system

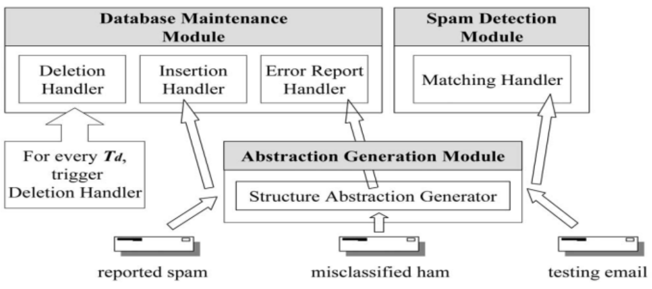


Fig. 7. System model of Cosdes.

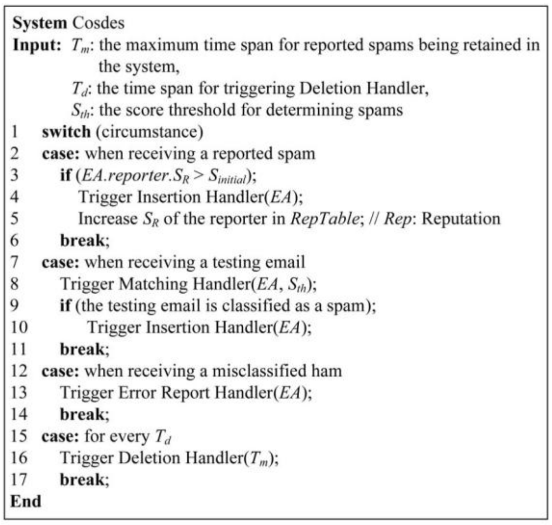
Cosdes to perform more robustly. We shall assess the effectiveness of our approach with real e-mail streams.

IV. COLLABORATIVE SPAM DETECTION SYSTEM COSDES

A complete collaborative spam detection system Cosdes is introduced in this section. The system model of Cosdes is given in Section 4.1. We then elaborate the processing handlers of Cosdes in Section 4.1. Finally, we describe the reputation mechanism of Cosdes in Section 4.3.

4.1 System Model of Cosdes

The system model of Cosdes is illustrated in Fig. 7, and the algorithmic form is outlined in Fig. 8. Initially, three parameters, Tm (the maximum time span for reported spams being retained in the system), Td (the time span for triggering Deletion Handler), and Sth (the score threshold for determining spams) should be given for Cosdes. Before starting to do the spam detection, Cosdes collects feedback spams for time Tm in advance to construct an initial database. Three major modules, Abstraction Generation Module, Database Maintenance Module, and Spam Detection Module, are



```
System Cosdes
Input: $T_m$: the maximum time span for reported spams being retained in
           the system,
       $T_d$: the time span for triggering Deletion Handler,
       $S_{th}$: the score threshold for determining spams
switch (circumstance)
case: when receiving a reported spam
  if ($EA.reporter.S_R > S_{initial}$);
    Trigger Insertion Handler($EA$);
    Increase $S_R$ of the reporter in $RepTable$; // $Rep$: Reputation
  break;
case: when receiving a testing email
  Trigger Matching Handler($EA$, $S_{th}$);
  if (the testing email is classified as a spam);
      Trigger Insertion Handler($EA$);
  break;
case: when receiving a misclassified ham
  Trigger Error Report Handler($EA$);
  break;
case: for every $T_d$
  Trigger Deletion Handler($T_m$);
  break;
End
```

Fig. 8. Algorithmic form of system Cosdes.

Included in Cosdes. With regard to Abstraction Generation Module, each e-mail is converted to an e-mail abstraction by Structure Abstraction Generator with procedure SAG. Three types of action handlers, Deletion Handler, Insertion Handler, and Error Report Handler, are involved in Database Maintenance Module. Note that although the term “database” is used, the collection of reported spams can be essentially stored in main memory to facilitate the process of matching. In addition, Matching Handler in Spam Detection Module takes charge of determining results.

4.2 Procedures of System Cosdes

In this section, we elucidate each procedure of system Cosdes. Cosdes deals with four circumstances by handlers (the algorithmic forms are shown in Fig. 9), and the detailed procedure flow will be explained as follows: For Insertion Handler in Fig. 9a, initially, the corresponding SpTree is found in SpTable according to the tag length of the inserted spam, and nowNode is assigned as the root of this SpTree. In lines 3-8, we iteratively insert the subsequences of the e-mail abstraction along the path from root to leaf. If nowNode is an internal node, the subsequence with 2i tags is inserted into level i, which is illustrated in Fig. 5. Meanwhile, the hash value of this subsequence is computed. Then, nowNode is assigned as the corresponding child node based on the type of the next tag. If the next tag is a start (end) tag, nowNode is assigned as the left (right) child node. Finally, when nowNode is processed to a leaf node, the subsequence with remaining tags is stored.

4.3 Reputation Mechanism

The principal concept of collaborative spam detection is to collect human judgment to block subsequent near-duplicate spams. To ensure the truthfulness of spam reports and to prevent malicious attacks, we propose the reputation mechanism to evaluate the credit of each reporter. The fundamental idea of the reputation mechanism is to utilize a reputation table to maintain a reputation score SR of each reporter according to the previous reliability record. Each inserted spam is given a suspicion score equal to SR of the reporter. In such a context, when doing near-duplicate detection, if the sum of suspicion scores of matched spams exceeds a predefined threshold, the testing e-mail will be classified as a spam.

V. PERFORMANCE EVALUATION

To assess the feasibility of system Cosdes,we conduct several experiments to explore its efficiency and detection results. The real spam data sets used in the experiments are from the e-mail servers of Computer Center in National Taiwan University, which has over 30,000 students. Since the ground truth of real e-mail streams is unavailable, spams are extracted from the well-known existing system, SpamAssassin. 3 Concerning hams, we not only include public data sets (around 4,000 e-mails) provided by SpamAssassin,4 but also obtain from volunteers. There are about 60,000 spams per day and a set of 7,000 or so hams in the data set. Note that numerous related works have evaluated the proposed methods with static databases.

5.1 Spammer Attack

To further verify the robustness of Cosdes, in this section, we simulate the spammer attack of random HTML tag insertion. We consider the situation that a spammer sends n identical e-mails at a time, where n is varied from 1,000 to 100,000. It is assumed that a sequence of random HTML tags is inserted into the beginning of each e-mail. The number of tags in a sequence is a random number between 1 and 50.5 Regarding the type of tag, we randomly choose them from HTML tag list.6 As shown in Fig. 16a, around 90 percent of emails are matched with other e-mails, meaning that only 10 percent of spams have completely distinct HTML tag sequences as random HTML tag insertion is applied. This is because the sequence preprocessing step of procedure SAG will delete nonempty tags that have no corresponding start tags or end tags. Random HTML tag insertion cannot generate legal tag sequences and thus most tags will be eliminated. Concerning the efficiency analysis, it can be observed in Fig. 16b that the sequence preprocessing step incurs very little overhead.

VI. CONCLUSION

In the field of collaborative spam filtering by near-duplicate detection, a superior e-mail abstraction scheme is required to more certainly catch the evolving nature of spams. Compared\ to the existing methods in prior research, in this paper, we explore a more sophisticated and robust e-mail abstraction scheme, which considers e-mail layout structure to represent e-mails. The specific procedure SAG is proposed to generate the e-mail abstraction using HTML content in e-mail, and this newly-devised abstraction can more effectively capture the near-duplicate phenomenon of spams. Moreover, a complete spam detection system Cosdes has been designed to efficiently process the near-duplicate

VII. ACKNOWLEDGMENTS

This work was supported in part by the National Science Council of Taiwan under Contracts NSC97-2221-E-002- 172-MY3.

REFERENCES

[1] E. Blanzieri and A. Bryl, "Evaluation of the Highest Probability SVM Nearest Neighbor Classifier with Variable Relative Error Cost," Proc. Fourth Conf. Email and Anti-Spam (CEAS), 2007.

[2] M.-T. Chang, W.-T. Yih, and C. Meek, "Partitioned Logistic Regression for Spam Filtering," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data mining (KDD), pp. 97-105, 2008.

[3] S. Chhabra, W.S. Yerazunis, and C. Siefkes, "Spam Filtering Using a Markov Random Field Model with Variable Weighting Schemas," Proc. Fourth IEEE Int'l Conf. Data Mining (ICDM), pp. 347-350, 2004.

[4] P.-A. Chirita, J. Diederich, and W. Nejdl, "Mailrank: Using Ranking for Spam Detection," Proc. 14th ACM Int'l Conf. Information and Knowledge Management (CIKM), pp. 373-380, 2005.

[5] R. Clayton, "Email Traffic: A Quantitative Snapshot," Proc. of the Fourth Conf. Email and Anti-Spam (CEAS), 2007.

[6] A.C. Cosoi, "A False Positive Safe Neural Network; The Followers of the Anatrim Waves," Proc. MIT Spam Conf., 2008.

[7] E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, "An Open Digest-Based Technique for Spam Detection," Proc. Int'l Workshop Security in Parallel and Distributed Systems, pp. 559-564, 2004.

[8] E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, "P2P-Based Collaborative Spam Detection and Filtering," Proc. Fourth IEEE Int'l Conf. Peer-to-Peer Computing, pp. 176-183, 2004.

[9] P. Desikan and J. Srivastava, "Analyzing Network Traffic to Detect E-Mail Spamming Machines," Proc. ICDM Workshop Privacy and Security Aspects of Data Mining, pp. 67-76, 2004.

[10] H. Drucker, D. Wu, and V.N. Vapnik, "Support Vector Machines for Spam Categorization," Proc. IEEE Trans. Neural Networks, pp. 1048-1054, 1999.

[11] D. Evett, "Spam Statistics," http:/