International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 4, June - 2012

# Detection of Spam through E-mail Abstraction Scheme

K.V.Srinivasa Rao[1]
*Associate Professor, Dept. of CSE, Prakasam Engg. College, Kandukur-5233105, AP., India*

S.Srinivasulu[2]
*HOD.Dept.of CSE Prakasam Engg. College, Kandukur-5233105 AP., India*

A.Amrutavalli[3]
*Dept. of CSE, Prakasam Engg. College, Kandukur-5233105, AP., India*

## Abstract

*In this paper we present a procedure to generate the email abstraction using HTML content in email and this newly devised abstraction can more effectively capture the near duplicate phenomenon of spams. The prior works mainly represent each email by a succinct abstraction derived from email content text. However these abstractions of emails cannot fully catch the evolving nature of spam's and are thus not effective enough in near duplicate detection. Moreover we represent each email using HTML tag sequence rather than content text. We design a complete Spam Detection System, which possesses an efficient near-duplicate matching and a progressive update scheme. This paper mainly focused on efficient similarity matching and reducing storage utilization.*

## 1. Introduction

Nowadays the email spam problem becomes more and more serious issue. Spam not only causes the misuse of time and computational resources, thus leading to financial losses, but it is also often used to advertise illegal goods and services or to promote online frauds. The most popular way of anti-spam detection is Spam filtering. A Spam filter is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to user's inbox. The primary challenge of spam detection lies in the fact that spammers will always find new ways to attack spam filters owing to the economic benefits of sending spam's. Spammers have no choice to but to send out large quantities of identical or similar spam's simultaneously to make profits. This specific feature of spam's can be designated as *near-duplicate phenomenon*, which is a significant key in the spam detection. **Definition 1 (Near-Duplicate).** *"Two e-mails are viewed as near-duplicate if their HTML tag sequences are exactly identical to each other."*

. The primary idea of the near-duplicate matching for spam detection is to block subsequent spams with similar content. The previous researchers have developed various methods on near-duplicate spam detection [5],[6],[9],[12],[15], these works are still subject to some drawbacks. Because these works mainly represent each e-mail by a succinct abstraction derived from e-mail content text. Moreover, hash based text representation is applied extensively. One major problem of these abstractions is that they may be too brief and thus may not be robust enough to withstand intentional attacks. The hash based text representation also suffers from the problem of not being suitable for all languages.

In this paper we explore to device a more sophisticated e-mail abstraction by using HTML content, which can more effectively capture the near duplicate phenomenon of spams. In this paper we propose the specific procedure Abstraction Generation 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 spam's. We devise an innovative tree structure, SpTrees, to store large amounts of the e-mail abstractions of reported spams. SpTrees contribute to the accomplishment of the efficient near-duplicate matching with a more sophisticated e-mail abstraction and we design complete spam detection system. Overall, there are three key points of this type of spam detection approach we have to be concerned about. First, an effective representation of e-mail (i.e., e-mail abstraction) is essential. Since a large set of reported spams has to be stored in the known spam database, the storage size of e-mail abstraction should be small. Moreover, the email mail abstraction should capture the near-duplicate phe- nomenon of spams, and should avoid accidental deletion of nonspam e-mails (also known as hams). Second, every incoming e-mail has to be matched with the large database, meaning that the near-duplicate matching process should be substantially efficient. Finally, the latest spams have to be included instantly and successively into the database so as to effectively block subsequent near-duplicate spams.

## 2. Abstraction Generation

We propose the specific procedure AG to generate the e-mail abstraction using HTML content in e-mail. The algorithmic form of Abstraction Generation is outlined in Fig. 1. Procedure AG is composed of three major phases, Tag Extraction Phase, Tag Reordering Phase, and <anchor> Appending Phase.

```
Procedure Abstraction Generation
I/P: E-Mail with html/text content-type
     The tag length threshold value of short email (Lth_short)
O/P: Email Abstraction (EA)
    1. Tag Extraction Phase
    2. Translate each tag into <tag.name>;
    3. Translate text into <mytext/>;
    4. Add all anchor tags to AnchorSet;
    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. Appending Phase
    13. If(EA.tag_length<Lth_short);
    14. Append AnchorSet in front of EA;
    15. Return EA;
End.
```

**Fig. 1. Algorithm for Abstraction Generation**

### Tag Extraction 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/>. <anchor> tags are then inserted into AnchorSet, and the first 1,023 valid tags are concatenated 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, we preprocess 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 following sequence of operations is performed in the preprocessing step.

1. Front and rear tags are excluded.
2. Nonempty tags that have no corresponding start tags or end tags are deleted. Besides, mismatched nonempty tags are also deleted.
3. All empty tags are regarded as the same and are replaced by the newly created <empty/> tag. Moreover, successive <empty/> tags are pruned and only one <empty/> tag is retained.
4. The pairs of nonempty tags enclosing nothing are removed.

### Tag Reordering Phase

On purpose of accelerating the near-duplicate matching process, we reorder the tag sequence of an e-mail abstraction in Tag Reordering Phase. Note that since the arrangement of HTML tags is regular and in pairs, various sequential patterns of tags are contained in e-mails. In the worst case, if we consider two e-mail abstractions which have the same tag length and differ only in their last tags, the difference cannot be detected until the last tags are compared. To handle this problem, we destroy the regularity by rearranging the order of tag sequence to lower the number of tag comparisons. Note that this process ensures that the newly assigned position numbers of e-mail abstractions with the same number of tags are completely identical. As such, the matching process can be accelerated without violating the definition of near-duplicate in this paper. In lines 8-11 of Fig. 1, each tag is assigned a new position number by function ASSIGN_PN (PN denotes for Position Number) with following expressions,

```
b=ceil (sqrt (L));
r= (PN_orig-1) %b,
q=floor ((PN_orig-1)/b) +1;
```
$PN_{new}= (b*r) + (b-q+1);$

The final e-mail abstraction is the concatenation of all tags with new position numbers.

### Tag Appending Phase

In this phase the tags in AnchorSet will be append in front of Email Abstraction whenever the tag length of an e-mail abstraction is smaller than a predefined tag length threshold of the short e-mail. The main objective of appending <anchor> tags is to reduce the probability that a ham is successfully matched with reported spams when the tag length of an e-mail abstraction is short.

## 3. Similarity Matching Process

In this paper we used SpTable and SpTrees for efficient matching process. 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. 2, 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 1, two e-mail abstractions are possible to be

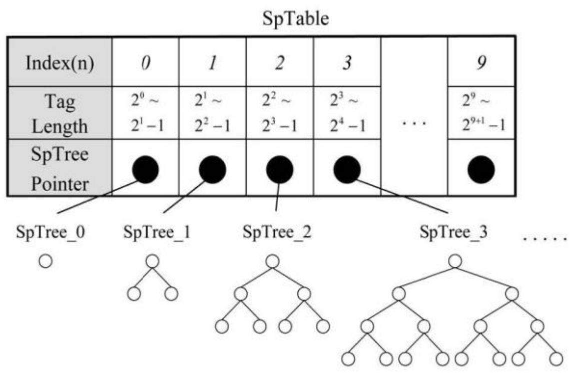

**Fig.2. SpTable and SpTrees**

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. 2, SpTable is created to record overall information of SpTrees. The $i_{th}$ column of SpTable links to the root of SpTree_i by a pointer, and e-mail abstractions with tag lengths ranging from $2^i$ to $2^{i+1}$ -1 belong to SpTree_i. An e-mail abstraction is segmented into several subsequences, and these subsequences are consecutively put into the corresponding nodes from low levels to high levels. As such, an e-mail abstraction is stored in one path from the root node to a leaf node of SpTree, and hence the matching between a testing e-mail and known spams is processed from root to leaf node. The primary goal of applying the tree data structure for storage is to reduce the number of tags required to be matched when processing from root to leaf. Since only subsequences along the matching path from root to leaf should be compared, the matching efficiency is substantially increased. To achieve efficient matching with balanced tree structure, SpTrees are designed to be binary trees. The branch direction of each SpTree is determined by a binary hash function. The hash function is defined as follows:

$$hash(seq)=f(seq[0])*2^{m-1}+f(seq[1])*2^{m-2}+...+f(seq[m-1])*2^{0}$$

where m is the number of tags in this subsequence and seq[n] denotes the tag type of the $n_{th}$ tag. The function f converts each type of tag to a unique integer. Moreover, for the subsequence which contains more than eight tags, we just use the first eight tags to generate the hash value (i.e., $m \leq 8$). With the hash function, most subsequence matching is transformed the integer matching, and hence the complexity of matching process can be substantially reduced. Moreover, with the hash function, the matching efficiency is substantially increased.

## 4. Spam Detection System model

The complete Spam Detection System is introduced here. Three major modules, Abstraction Generation Module, Database Maintenance Module, and Spam Detection Module are included in our system. In Abstraction Generation Module, each e-mail is converted to an e-mail abstraction by Structure Abstraction Generator with Abstraction Generation

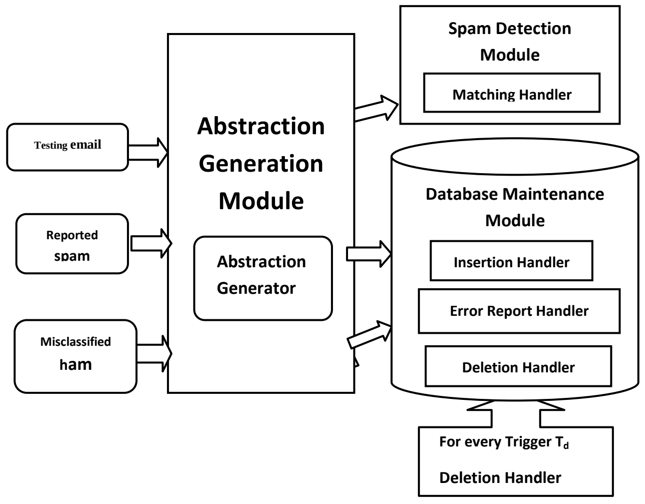

**Fig.3. Spam Detection System Model**

procedure. 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 spam’s 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. There are three types of e-mails, reported spam, testing e-mail, and misclassified ham, required to be dealt with by Spam Detection System. The algorithmic form is outlined in Fig.4. 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 the system.

**Algorithm for Spam Detection System:**

```
Input: $T_m$: The maximum time span
       $T_d$: The time span for Deletion Handler
       $S_{th}$: The score threshold for determining spams
  1.  Begin switch(c)   //circumstance(c)
  2.  case 1: when receiving reported spam
  3.          if(EA.reporter.Sr>$S_{initial}$)
  4.             Insertion Handler(EA);
  5.             $S_r = S_r + 1$;
  6.          End if;
  7.   break;
  8.  case 2: when receiving a testing e-mail
  9.             Matching Handler(EA,$S_{th}$);
  10.               if(testing email= = spam mail)
  11.                  Matching Handler(EA);
  12.             End if;
  13.  break;
  14. case 3: when receiving a misclassified ham
  15.            Error Report Handler(EA);
  16.   break;
  17. case 4: for every  Td
  18.           Deletion Handler(Tm);
  19.  break;
  20. End switch;
End;
```

**Fig: 4. Algorithmic form of system model**

Whenever our system receiving a reported spam, **Insertion Handler** adds the e-mail abstraction of this spam into the database. The algorithmic form of insertion handler is as follows.

**Algorithm for Insertion Handler:**

```
Procedure of Insertion Handler
I/P: EA: Email Abstraction
1.  Find the SpTree from SpTable according to
    EA.tag_length;
2.  Assign newnode to SpTree root node
3.  For (i=0 to SpTree.height)
4.  If(newnode is not a leaf node)
5.    add the subsequences with $2^i$ tags;
6.  //Compute the hash value of this subsequence;
7.     hash_Value()
8.      {
9.       hash= hash + (seq* Math.pow(2, ind - 1));
10.      hash_List.add(hash);
11.     }
12.  newnode=the corresponding childnode;
13. End if;
14. Else
15.     Insert the subsequence with remaining tags;
16.     Compute hash_Value() of this subsequence;
End
```

**Fig.5. Algorithmic form of Insertion Handler**

In fig.5, initially, the corresponding SpTree is found in SpTable according to the tag length of the inserted spam, and newNode is assigned as the root of this SpTree. In lines 3-11, we iteratively insert the subsequences of the e-mail abstraction along the path from root to leaf. If newNode is an internal node, the subsequence with $2^i$ tags is inserted into level i. Meanwhile, the hash value of this subsequence is computed. Then, newNode is assigned as the corresponding child node based on the type of the next tag. If the next tag is a start (end) tag, newNode is assigned as the left (right) child node. Finally, when newNode is processed to a leaf node, the subsequence with remaining tags is stored.

Whenever a new testing e-mail arrives, **Matching Handler** performs the near-duplicate detection with collected spams to do the judgment. Meanwhile, if a testing e-mail is classified as a spam, this e-mail will be viewed as a reported spam and be added into the database. Matching Handler (as shown in Fig. 6) is the most significant procedure in our system to achieve efficient matching between every testing e-mail and the known spam database. There are two major phases in the matching process: Approximate Matching Phase and Exact Matching Phase. As mentioned in Section 3 the tag lengths of e-mail abstractions in an SpTree may not be identical. However, two e-mail abstractions are possible to be near-duplicate only when the numbers of their tags are identical. For this reason, in Approximate Matching Phase, we traverse directly to the targeted leaf node based on the types of tags at positions $2^i$ without doing tag comparisons. It is certain that a testing e-mail may merely be near-duplicate with spams which have the same tag length and are in the same path. Therefore, we tentatively record the information of spams, which appear in the targeted leaf node and have the same tag length, into a candidate set candSet. The main objectives of the approximate matching are: 1) to reduce unnecessary tag comparisons of e-mails with different tag lengths, and 2) to exclude e-mails which can be determined without the exact tag matching. Subsequently, the process starts Exact Matching Phase from the root of the SpTree. For each level, in lines 11-17, the hash values of subsequences are matched first. Then, we do the exact matching of subsequences only if their hash values are matched. The unmatched information of spams will be deleted from candSet.

**Algorithm for Matching Handler:**

```
Procedure of Matching Handler
I/P: EA: the email abstraction of a testing email
     S_th: the score threshold for determining spams
O/P: the detection result
  1.  var final;//the final level which exact matching is processed
  2.  var candSet;//the set for tentative info of candidate spams
  3.  //Approximate Matching Phase
  4.  Find the corresponding SpTree in SpTable with EA.tag_length;
  5.  Traverse directly to the targeted leaf node;
  6.  for(each subsequence in the leaf node)
  7.     If(EA.tag_length==subsequence.tag_length)
  8.      candSet.insert(subsequence.info);
  9.  End loop;
  10. //Exact Matching Phase
  11. Insert the newNode as SpTree.root;
  12. for(i=0 to final)
  13.  for(each subseq in candSet)
  14. If(subseq.hashvalue==EA.current_subseq.hashvalue )
  15. If (subseq!=EA.current_subseq.hash_value )
  16. candSet.delete(subseq.info);
  17. else candSet.insert(subseq.info);
  18. new node =the corresponding child node ;
  19. sum=the sum of Sr of all candidate spams in candSet;
  20. if (sum>S_th)   return spam;
  21. else            return ham;
End
```

**Fig. 6. Algorithmic form of Matching Handler**

Moreover, we design that the exact tag matching is only processed to level final, only the first $2^{f\_level+1}$-1 tags are exactly matched. It means that the looser matching criterion is applied when the length of an e-mail abstraction is longer than $2^{f\_level+1}$. This looser criterion substantially promotes the efficiency of matching but does not influence detection results owing to the effects of the preceding approximate matching and the tag reordering process of procedure SAG. Finally, if the sum of $S_R$ of all candidate spams in candSet exceeds $S_{th}$, the testing e-mail will be classified as a spam.

.**Algorithm for Error Report Handler:**

```
Procedure of Error Report Handler
I/P: The EA of misclassified ham
  1. Find the corresponding SpTree in SpTable with EA.tag_length;
  2. Perform Matching Handler();
  3. Reset Sr of the matched spams as 0;
  4. Update Sr of related reporters in RepTable;
End
```

**Fig.7. Algorithmic form of Error Report Handler**

When receiving a misclassified ham, Error Report Handler (shown in Fig. 7) first finds the corresponding SpTree and does the matching process as the same in Matching Handler. For the spams matched with the reported misclassified ham, we reset $S_R$ of these spams as 0 to avoid subsequent misclassification incurred by the identical group of spams. In addition, the reputation scores of reporters who cause the false positive error are halved to prevent continuous attacks by specific users.

**Algorithm for Deletion Handler:**

```
Procedure of Deletion Handler
I/P: Tm: the maximum time span for reported spams..
  1. Var   cTime,Ts;//ts:timespan,cTime:currentt (i=0 to SpTree)
  2.   for(each node in the SpTree in inorder)
  3.   for(each subseq in the node )
  4.    if (cTime-suseq.Ts>Tm)
  5.     delete the subsequence;
  6.    End if;
  7.  End loop;
End.
```

**Fig.8. Algorithmic form of Deletion Handler**

Moreover, to delete obsolete spams, for every Td, Deletion Handler (as shown in Fig. 8) traverses each SpTree in inorder (traverses the left subtree, visit the root, and then traverses the right subtree) to visit all nodes in SpTrees. For each subsequence, if the existing time exceeds Tm, it will be viewed as outdated and be deleted from this node. As such, all obsolete spams are removed from the known spam database after Deletion Handler is processed.

### 4.1. Reputation Mechanism

The principal concept of 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 $S_R$ of each reporter according to the previous reliability record. Each inserted spam is given a suspicion score equal to $S_R$ 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. The reputation mechanism is described in detail as follows:

1. Each reporter is assigned an initial score $S_{initial}$ when he submits a reported spam at the first time.
2. If a reporter submits any feedback spam once more, the reputation score will be incremented by a smaller incremental score $S_{incre}$ . The value of Sincre is set as $S_{intial}/10$ in the experiments.
3. If a reporter is charged that his previous feedback spam is mistaken, the reputation score will be halved.

To prevent malicious error reports and to attain a near-zero false positive rate, we cautiously increase the reputation score but drop it drastically while a false positive error is issued. On the other hand, when $S_R$ of a reporter is smaller than $S_{initial}$, his subsequent feedback spams will not be added into the database until $S_R$ is equal to or larger than $S_{initial}$. Regarding the parameter $S_{th}$, we simply use a fixed small value (set as three in the experiments) instead of determining the threshold according to the ratio of total users. The reason is that as long as there are certain trusty users reporting the e-mails with the same e-mail abstraction as spams, it is sufficiently reliable to classify the subsequent near-duplicate e-mails as spams.

## 5. Experimental results

We conduct the efficiency investigation of Spam Detection System on inserting e-mail abstractions into the database and deleting outdated spams from the database. we only study the performance of Insertion Handler and Deletion Handler in our system. Fig. 9a shows the execution time of Insertion Handler of our system with the number of e-mails varied. The execution time grows linearly and costs merely 3.5 seconds for inserting 100,000 spams into the database. On the other hand, the performance of Deletion Handler is shown in Fig. 9b.

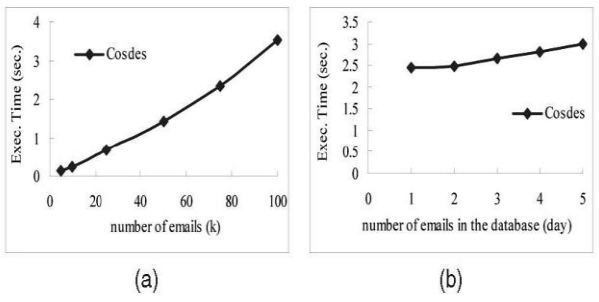

**Fig.9. Performance of Insertion Handler and Deletion Handler in spam detection system.(a) Excution time of insertion mails. (b)Excution time of Deletion mails.**

We evaluate the execution time of deleting spams in one day while the number of e-mails in the database varied. The main purpose of this experiment is to examine whether the efficiency of deletion will be influenced by the amount of e-mails stored in SpTrees. It is shown that the deletion process costs only 2 to 3 seconds in each situation, and the execution time slightly increases with the amount of e-mails. Therefore, we can observe that both the processes of insertion and deletion in our system are efficient.

## 6. Conclusion

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 for Abstraction Generation 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. In this paper we used SpTable and SpTrees for efficient matching process. Moreover, a complete spam detection system been designed to efficiently process the near-duplicate matching and to progressively update the known spam database.

## 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] A.C. Cosoi, “A False Positive Safe Neural Network; The Followers of the Anatrim Waves,” Proc. MIT Spam Conf., 2008.
[5] 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.
[6] 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
[7] A. Gray and M. Haahr, “Personalised, Collaborative Spam Filtering,” Proc. First Conf. Email and Anti-Spam (CEAS), 2004.

[8] A. Kolcz, A. Chowdhury, and J. Alspector, “The Impact of Feature Selection on Signature-Driven Spam Detection,” Proc. First Conf. Email and Anti-Spam (CEAS), 2004.
[9] J.S. Kong,P.O.Boykin, B.A. Rezaei, N.Sarshar, and V.P. Roychowdhury, “Scalable and Reliable Collaborative Spam Filters: Harnessing the Global Social Email Networks,” Proc. Second Conf. Email and Anti-Spam (CEAS), 2005.
[10] T.R. Lynam and G.V. Cormack, “On-Line Spam Filter Fusion,” Proc. 29th Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 123-130, 2006.
[11] I. Rigoutsos and T. Huynh, “Chung-Kwei: A Pattern-Discovery-Based System for the Automatic Identification of Unsolicited E-Mail Messages (SPAM),” Proc. First Conf. Email and Anti-Spam (CEAS), 2004.
[12] S. Sarafijanovic and J.-Y.L. Boudec, “Artificial Immune System for Collaborative Spam Filtering,” Proc. Second Workshop Nature Inspired Cooperative Strategies for Optimization (NICSO), 2007.
[13] S. Sarafijanovic, S. Perez, and J.-Y.L. Boudec, “Improving Digest-Based Collaborative Spam Detection,” Proc. MIT Spam Conf., 2008.
[14] S. Sarafijanovic, S. Perez, and J.-Y.L. Boudec, “Resolving FP-TP Conflict in Digest-Based Collaborative Spam Detection by Use of Negative Selection Algorithm,” Proc. Fifth Conf. Email and Anti-Spam (CEAS), 2008.
[15] K. Yoshida, F. Adachi, T. Washio, H. Motoda, T. Homma, A. Nakashima, H. Fujikawa, and K. Yamazaki, “Density-Based Spam Detector,” Proc. 10th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD), pp. 486-493, 2004.
[16] F. Zhou, L. Zhuang, B.Y. Zhao, L. Huang, A.D. Joseph, and J.D. Kubiatowicz, “Approximate Object Location and Spam Filtering on Peer-to-Peer Systems,” Proc. ACM/IFIP/USENIX Int’l Middle-ware Conf., pp. 1-20, 2003.
[17] K.M. Schneider, “Brightmail URL Filtering,” Proc. MIT Spam Conf., 2004.
[18] D. Sculley and G.M. Wachman, “Relaxed Online SVMs for Spam Filtering,” Proc. 30th Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 415-422, 2007.
[19] C.-Y. Tseng, J.-W. Huang, and M.-S. Chen, “Promail: Using Progressive Email Social Network for Spam Detection,” Proc. 10th Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), pp. 833-840, 2007.
[20] Z. Wang, W. Josephson, Q. Lv, and K.L.M. Charikar, “Filtering Image Spam with Near-Duplicate Detection,” Proc. Fourth Conf. Email and Anti-Spam (CEAS), 2007.

## Authors Profile

**1.K.V.Srinivasa Rao** currently working as an Associate professor in the department of Computer Science and Engineering, at Prakasam Engineering College, Kandukur, A.P. India. He is having 3 years of research and 16 years of teaching experience. He is a research scholar in the department of CSE at Acharya Nagarjuna University, India.
E-mail: srinivasa_rao_kalva@yahoo.co.in

**2. S. Srinivasulu** currently working as a head of the Computer Science department at Prakasam Engineering College, Kandukur, A.P. India. He is having 4 years of research and 16 years of teaching experience. He has published several research papers in various peer reviewed International Journals. He is a research scholar in the department of CSE at JNTUH University, India.
E-mail:sreenivasulusadineni@gmail.com

**3. Kum.A.Amrutavalli** M.Tech (CSE) from Prakasam Engineering College, Kandukur, Prakasam (Dt.), Affiliated by JNTUK, Kakinada, A.P., India.
E-mail:amrutavallia@gmail.com