IJCSET |October 2012 | Vol 2, Issue 10, 1447-1451 www.ijcset.net | ISSN:2231-0711

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

I.Kalpana , B.Venkateswarlu

Avanthi Institute of Engineering & Technology, Visakhapatnam.

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.

INTRODUCTION

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 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 Although the techniques used by spammers vary constantly, there is still one enduring feature: spams with identical or similar content are sent in large quantities and successively. Since only a small amount of e-mail users will order products or visit websites advertised in spams, spammers have no choice but to send a great quantity of spams to make profits. It means that even with developing and employing unexpected new tricks, spammers still have to send out large quantities of identical or similar spams simultaneously and in succession. This specific feature of spams can be designated as the near-duplicate phenomenon, which is a significant key in the spam detection problem In view of above facts, the notion of collaborative spam filtering with near-duplicate similarity matching scheme has recently received much attention. The primary idea of the near-duplicate matching scheme for spam detection is to maintain a known spam database, formed by user feedback, to block subsequent spams with similar content. Collaborative filtering indicates that user knowledge of what spam may subsequently appear is collected to detect following spams. 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 abstraction should capture the near-duplicate phenomenon of spams, and should avoid accidental deletion of non spam 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 Although previous researchers have developed various methods on near-duplicate spam detection , these works are still subject to some drawbacks. To achieve the objectives of small storage size and efficient matching, prior 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. A common attack to this type of representation is to insert a random normal paragraph without any suspicious keywords into unobvious position of an e-mail. In such a context, if the whole e-mail content is utilized for hash based representation, the near-duplicate part of spams cannot be captured. In addition, the false positive rate (i.e., the rate of classifying hams as spams) may increase because the random part of e-

mail content is also involved in e-mail abstraction. On the other hand, hash-based text representation also suffers from the problem of not being suitable for all languages. Finally, images and hyperlinks are important clues to spam detection, but both of them are unable to be included in hash-based text representation.

We explore to devise a more sophisticated email abstraction, which can more effectively capture the near duplicate phenomenon of spams. Motivated by the fact that email users are capable of easily recognizing similar spams by observing the layouts of e-mails, we attempt to represent each e-mail based on the e-mail layout structure. Fortunately, almost all e-mails nowadays are in Multipurpose Internet Mail Extensions (MIME) format with the text/html content type. That is, HTML content is available in an e-mail and provides sufficient information about e-mail layout structure. In view of this observation.

Existing system:

Various methods on near-duplicate spam detection have been developed. These works are still subject to some drawbacks. To achieve the objectives of small storage size and efficient matching, prior 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. A common attack to this type of representation is to insert a random normal paragraph without any suspicious key-words into unobvious position of an e-mail. In such a context, if the whole e-mail content is utilized for hash-based representation, the near-duplicate part of spams cannot be captured. In addition, the false positive rate (i.e., the rate of classifying hams as spams) may increase because the random part of e-mail content is also involved in e-mail abstraction. On the other hand, hash-based text representation also suffers from the problem of not being suitable for all languages. Finally, images and hyperlinks are important clues to spam detection, but both of them are unable to be included in hash-based text representation.

Proposed System:

In this paper, we design a complete spam detection system Collaborative Spam Detection System (Cosdes). Cosdes possesses an efficient near-duplicate matching scheme and a progressive update scheme. The progressive update scheme not only adds in new reported spams, but also removes obsolete ones in the database. With Cosdes maintaining an up-to-date spam database, the detection result of each incoming e-mail can be determined by the near-duplicate similarity matching process. In addition, to withstand intentional attacks, a reputation mechanism is also provided in Cosdes to ensure the truthfulness of user feedback.

The contributions of this paper are as follows:

1. We propose the specific procedure SAG 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.
2. 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.
3. We design a complete spam detection system Cosdes with 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

I ABSTRACTION GENERATION

In this module we generate an email abstraction. Here we use SAG (Structure Abstraction Generation) procedure to generate the email abstraction. First read html/text content type based input mail. This module composed of three major phases.

1. Tag Extraction phase:

In this phase we read input mail and get the each and tags. Transform each text into <mytext/> tag, add all the anchor tag and add the remaining tags. Preprocess the tag sequence.

2. Tag Reordering Phase:

In this phase we reorder each and every tag. Assign the position number. Add all the tags with the position number (EA).

3. Appending Phase:

Append the anchor set in front of EA.H355344

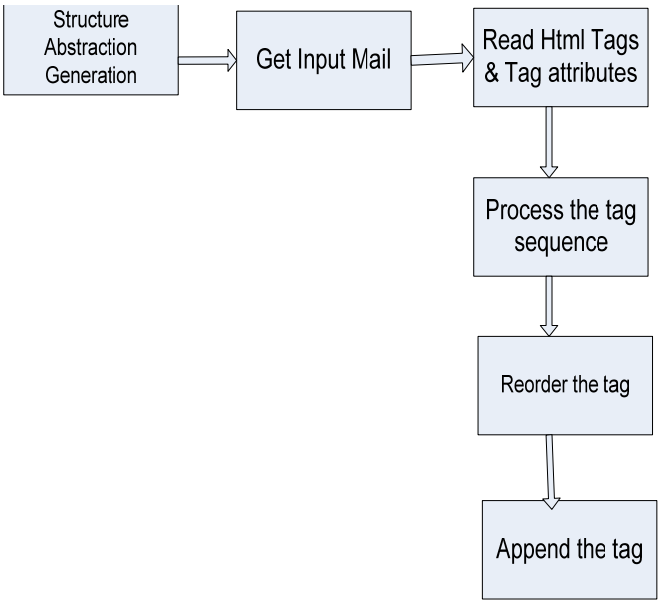

Fig: Tag Extraction Phase

II SAG STRUCTURED ABSTRACTION GENERATION:

This algorithm is used to generate the e-mail abstraction using HTML content in e-mail. It 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/>. <anchor> tags are then inserted into AnchorSet, and the first 1,023 valid tags are concatenated to form the tentative e-mail abstraction.

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.

```
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. Algorithmic form of system Cosdes.

On purpose of accelerating the near-duplicate matching process, we reorder the tag sequence of an e-mail abstraction in Tag Reordering Phase. 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.

III COSDES SYSTEM:

Cosdes deals with four circumstances by handlers. When receiving a reported spam it handles insertion. When receiving a testing mail it detect whether the mail is spam or ham. When receiving misclassified ham it handles the error report handler.

```
System Cosdes
Input: Tm: the maximum time span for reported spams being retained in
           the system,
       Td: the time span for triggering Deletion Handler,
       Sth: the score threshold for determining spams
1  switch (circumstance)
2  case: when receiving a reported spam
3     if (EA.reporter.SR > Sinitial);
4        Trigger Insertion Handler(EA);
5        Increase SR of the reporter in RepTable; // Rep: Reputation
6     break;
7  case: when receiving a testing email
8     Trigger Matching Handler(EA, Sth);
9     if (the testing email is classified as a spam);
10       Trigger Insertion Handler(EA);
11    break;
12 case: when receiving a misclassified ham
13    Trigger Error Report Handler(EA);
14    break;
15 case: for every Td
16    Trigger Deletion Handler(Tm);
17    break;
End
```

Fig. Algorithmic form of system Cosdes

IV INSERTION HANDLER:

In the insertion handler 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. Insert the subsequences of the e-mail abstraction along the path from root to leaf. If nowNode 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, 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.

```
Procedure of Insertion Handler
Input: EA: the email abstraction required to be inserted
1  Find the corresponding SpTree in SpTable according to
   EA.tag_length;
2  nowNode = SpTree.root;
3  for (i = 0 to SpTree.height)
4     if (nowNode is not a leaf node)
5        Insert the subsequence with 2^i tags;
6        Compute the hash value of this subsequence;
7        nowNode = the corresponding child node;
8     else // nowNode is a leaf node
9        Insert the subsequence with remaining tags;
10       Compute the hash value of this subsequence;
End
```

Fig. Algorithmic form of Insertion Handler

V . MATCHING HANDLER

```
Procedure of Matching Handler
Input: EA: the email abstraction of a testing email,
       Sth: the score threshold for determining spams
Output: the detection result
1  var f_level; // 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 based on the types of tags
   at positions 2^i;
6  for (each subsequence in the leaf node)
7     if (EA.tag_length == subsequence.tag_length)
8        candSet.insert (subsequence.info);
9  // Exact Matching Phase
10 nowNode = SpTree.root;
11 for (i = 0 to f_level)
12    for (each subsequence in candSet)
13       if (subsequence.hash_value ==
            EA.current_subsequence.hash_value)
14          if (subsequence!= EA.current_subsequence)
15             candSet.delete (subsequence.info);
16       else candSet.delete (subsequence.info);
17    nowNode = the corresponding child node;
18 sum = the sum of SR of all candidate spams in candSet;
19 if (sum > Sth) return spam;
20 else           return ham;
End
```

Fig. Algorithmic form Matching Handler

Matching Handler is the most significant procedure in Cosdes 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. 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 2i 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, 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. Moreover, we design that the exact tag matching is only processed to level f level, namely, 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 SR of all candidate spams in candSet exceeds Sth, the testing e-mail will be classified as a spam.

VI ERROR REPORT HANDLER

When receiving a misclassified ham, Error Report Handler 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 SR 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.

```
Procedure of Error Report Handler
Input: EA: the email abstraction of a misclassified ham
1  Find the corresponding SpTree in SpTable with EA.tag_length;
2  Do the matching as the same in Matching Handler;
3  Reset SR of the matched spams as 0;
4  Update SR of related reporters in RepTable;
End
```

Fig. Algorithmic form of Error Handler

VII 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 matching and to progressively update the known spam database. Consequently, the most up-to-date information can be invariably kept to block subsequent near-duplicate spams. In the experimental results, we show that Cosdes significantly outperforms competitive approaches, which indicates the feasibility of Cosdes in real-world applications.

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://spam-filter-review.topten reviews.com/spam-statistics.html, 2006.
[12] A. Gray and M. Haahr, "Personalised, Collaborative Spam Filtering," Proc. First Conf. Email and Anti-Spam (CEAS), 2004.
[13] S. Hershkop and S.J. Stolfo, "Combining Email Models for False Positive Reduction," Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 98-107, 2005.

AUTHORS:

Mr. B. Venkateswarlu has completed his Bachelor of Technology of CSE at Andhra University. He is completed M.Tech (CSE) at Andhra University, Guntur, Andhra Pradesh. He has 16 Yrs of experience in teaching field, presently working as Associate Professor in Computer Science & Engineering at Avanthi Institute of Engineering and Technology, Visakhapatnam, Andhra Pradesh.

Ms. I. Kalpana has pursuing her M.Tech (CSE) at Avanthi Institute of Engineering and Technology, Visakhapatnam, Andhra Pradesh, she is doing her research work in Data mining.