SpamGuru: An Enterprise Anti-Spam Filtering System

Richard Segal, Jason Crawford, Jeff Kephart, Barry Leiba

IBM Thomas J. Watson Research Center
{rsegal, ccjason, kephart, barryleiba}@us.ibm.com

Abstract. Spam-reduction techniques have developed rapidly over the last few years, as spam volumes have increased. We believe that no one anti-spam solution is the "right" answer, and that the best approach is a multi-faceted one, combining various forms of filtering with infrastructure changes, financial changes, legal recourse, and more, to provide a stronger barrier to spam than can be achieved with one solution alone. SpamGuru addresses the part of this multi-faceted approach that can be handled by technology on the recipient's side, using plug-in tokenizers and parsers, plug-in classification modules, and machine-learning techniques to achieve high hit rates and low false-positive rates.

1 Introduction

Spam-reduction techniques have developed rapidly over the last few years, as spam volumes have increased. We believe that the spam problem requires a multi-faceted solution that combines a broad array of filtering techniques with various infrastructural changes, changes in financial incentives for spammers, legal approaches, and more [1]. This paper describes one part of a more comprehensive anti-spam research effort undertaken by us and our colleagues: SpamGuru, a collaborative anti-spam filter that combines several learning, tokenization, and user interface elements to provide enterprise-wide spam protection with high spam detection rates and low false-positive rates.

Three basic design principles underlie SpamGuru. First, it simplifies administration. The infrastructure and productivity costs of spam in today's corporate environment are staggering. These costs are compounded by the time and effort system administrators must devote to blocking spam and preventing false positives. SpamGuru automates many decisions and tasks (such as maintaining black- and white-lists), thereby reducing I/T costs. Second, SpamGuru is highly customizable and tunable by both system administrators and individual users. No two organizations are alike; each has its own preferences regarding false-positive rate, user control, and other policy-related settings. Similarly, users differ in their definition of spam, and in their toleration of it. Giving users the ability to customize e-mail filtering can improve user satisfaction and eliminate support issues created by a one-size-fits-all solution. Third, by combining multiple classifiers and tokenization methods, SpamGuru provides a very low false positive rate that can be tuned to suit the administrator's or user's spam detection vs. false positive tradeoff. In addition to providing superior performance, the multiple classifier approach also makes SpamGuru robust against changes in spammer tactics.

This paper highlights some of the techniques that we use to achieve these three design goals. The next section provides an overview of SpamGuru and how it integrates into an organization's e-mail infrastructure. The following section details each of SpamGuru's filtering technologies and how they are combined to form a single aggregate filter. Then, we describe experiments that evaluate SpamGuru's filtering technologies individually and in combination with one another in order to assess the efficacy of the multiple classifier approach. The paper finishes with a discussion of related work and a summary of our main results.

2 System Overview

Fig. 1 provides an overview of our anti-spam architecture. Mail enters the enterprise via an SMTP server. Mail is relayed from the SMTP server to the SpamGuru anti-spam server. SpamGuru analyzes each incoming message and then assigns a score from 0 to 1000 for each user that is to receive that message, with higher scores indicating a higher likelihood that the message is spam. The message is then routed to the appropriate destination based on system and user preferences. If the message is identified as legitimate e-mail, it is delivered to the user's mailbox

untouched. If the message is deemed spam, the system does one of four actions based on system- and user-defined preferences: the message may be deleted, filed in an archive of recently blocked e-mail, sent to a challenge queue that provides a challenge/response verification of sender identity, or labeled as probable spam and delivered to the client for the client to take its own action as appropriate. Each approach has its good and bad aspects. For example, deletion saves valuable disk space and other resources, but makes it impossible to retrieve mail in the event of false positives. The recipient's mail client may do further spam analysis, but such client-side analysis would happen outside of SpamGuru, which is purely server-based.

An important component of our overall architecture is user reporting of junk and good e-mail, a process we call "voting". Voting is at present the main source of input to our learning algorithms. To make user voting as easy as possible, we have extended the Lotus Notes mail client to include buttons for instant reporting of spam and non-spam messages. In addition, we have added to the client the two folders "Junk Mail" and "Borderline Messages". The "Junk Mail" folder is common to many anti-spam solutions and is manly a mechanism for reducing the impact of false positives. Mail that is labeled as probably spam can be placed in a user's Junk Mail folder, giving the user a second chance to peruse the junk mail folder to identify any messages that were incorrectly branded as spam.

The problem with Junk Mail folders is, naturally, that they contain a lot of junk. Users have little incentive to frequently scan a junk mail folder containing hundreds of messages, especially when using an anti-spam solution with a low false positive rate. SpamGuru's "Borderline Messages" folder helps alleviate this problem. The Borderline Messages folder is designed to contain a small set of messages for which the anti-spam server is relatively uncertain in its classification, receiving both borderline spam and borderline good mail. The inclusion of borderline good messages encourages users to view their borderline folder and to vote mail as either good or spam by threatening to delete mail from the folder after a specified time period. However, removing borderline good e-mail could cause the same problem we are trying to prevent: the user losing important e-mail due to false positives. SpamGuru places borderline good messages in both the user's inbox and the user's borderline messages folder. Thus users may periodically check for misclassified mail without reducing the system's perceived effectiveness.

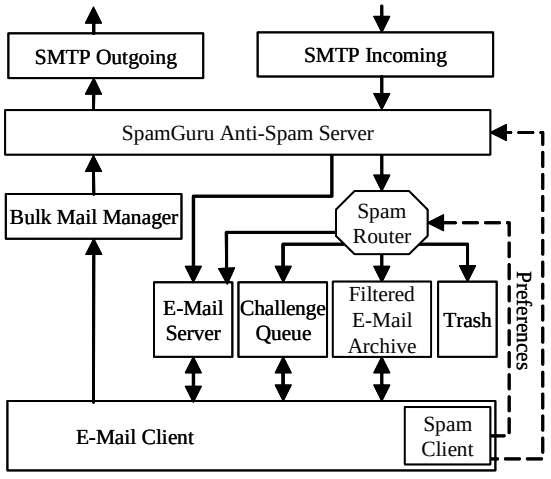

Fig. 1: The SpamGuru anti-spam architecture.

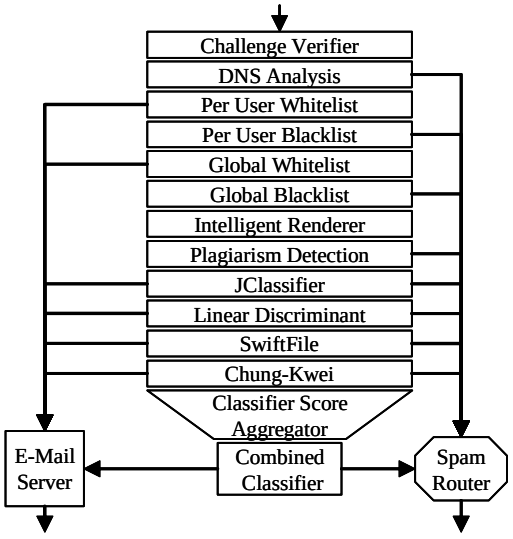

Fig. 2: The SpamGuru Server filtering pipeline

While many anti-spam solutions only consider incoming mail, the analysis of outgoing mail is also valuable. Outgoing mail is a good resource for learning both personal and global white-lists. Also, senders of legitimate bulk e-mail must comply with a variety of local, national, and international laws while honoring the privacy and the preferences of their customers. SpamGuru is integrated with IBM's Global Email Cleansing Service (GECS) to help prevent legitimate bulk e-mailers from becoming part of the problem [2].

One of the challenges faced by anti-spam filters is that no two users define spam in the same way. While one person may consider the constant barrage of mortgage offers problematic, another user may see an opportunity to save on a new home. Most enterprise spam filters use a single, necessarily cautious definition of spam for filtering mail for all users. An alternative frequently advocated in the literature is personalized classifiers tuned to individual preferences [3][4]. Personalized classifiers learn by asking users to label wanted and unwanted mail by reporting any

mistakes it finds using a “Report Spam” or “Report Good” button. Personal classifiers place the burden of providing training data on the user and cannot take advantage of the collaborative efforts of others within their organization. SpamGuru addresses this issue by creating both a global classifier and a personalized classifier for each user. The results of each classifier are dynamically combined to benefit from global learning while respecting individual preferences. While arguments can be made for situating personal classifiers on the client, we have implemented SpamGuru as a pure server-based solution in which personalized models are stored and used on the server. This choice makes SpamGuru easier to deploy and maintain in a large enterprise environment.

3 Filtering Pipeline

SpamGuru’s anti-spam filter design supports the three principles set forth in the introduction. It enables easy administration and configurability of large-scale enterprise anti-spam systems, and it yields a high performance, robust filter that achieves a high level of throughput on modest hardware. The filter architecture, a realization of which is illustrated in Fig. 2, is a pipeline of anti-spam filter components (classifiers, rendering algorithms, tokenizers and other spam-identification tools) through which the e-mail messages are routed. Each filter component uses common generic interfaces that enable one to combine the components in essentially any order, resulting in an aggregate filter that is more effective than any individual classifier. The combination of several different filters in the pipeline also leads to greater robustness, because any new spamming technique is unlikely to evade all of the filters simultaneously. The flexibility offered by the common interfaces is important because the spam problem is constantly evolving, and because each enterprise has its individual characteristics that may favor one configuration over another. Thus administrators need the ability to experiment with and change configurations. In future releases, we intend to make the classification interface public so that we can support third-party and user-provided classifiers.

The pipeline processes an incoming e-mail message by passing it along from one filter component to the next. Each classifier analyzes the message and assigns it a score based on the classifier’s prediction of how likely it is to be spam. For each classifier, we define two thresholds. The threshold $T_{DefiniteSpam}$ indicates the score above which the classifier is so certain that the message is spam that the message can be routed directly to the spam router with no need for further downstream processing. Similarly, the threshold $T_{DefiniteGood}$ indicates the score below which the classifier is so certain that the message is *not* spam that the message can be routed directly to the mail server. Messages between these two thresholds are annotated with the classifier’s score and other information that can be used as features by downstream classifiers) and passed to the next classifier. If a message reaches the end of the pipeline without being declared definite spam or definite good mail, the scores of all the classifiers are combined to produce a single aggregate score that is thresholded to determine the message’s classification, (and hence destination).

This short-circuit mechanism reduces the average number of classifiers needed to process each message, helping to make SpamGuru’s pipeline computationally efficient. The exact placement of the various classifiers can have an important bearing on efficiency; Fig. 2 represents an arrangement that we have found to work well in practice. The basic principle is to place early in the pipeline the filters that are least expensive and/or maximally discriminant (i.e. ones that can with confidence brand a message as definitely spam or definitely good). Thus simple approaches that only analyze headers are placed earlier than more computationally expensive algorithms that filter on message content, and message-content filters with very low false-positive rates are placed earlier than the others.

The computational cost of receiving, parsing, rendering, and tokenizing e-mail often dominates that of running the individual classifiers. SpamGuru reduces computational costs by performing rendering once, and by sharing tokenization across different classifiers wherever possible. Consequently, SpamGuru’s computational cost is only slightly more than that of running a single classifier, and as we will show the benefits in terms of improved classification accuracy are considerable. Further computational savings will be realized in future versions of SpamGuru by exploiting similar data structures and analysis code that exist in several classifiers, such as JClassifier, SwiftFile, and Linear Discriminant.

In the current instantiation of the pipeline, displayed in Fig. 2, the first component is the challenge and payment verification engine. This engine classifies incoming e-mail as good if it contains a valid challenge response or payment stamp. This component is still under development, but we are in the process of implementing the Charity Seals method of charging to send e-mail to recipients for which the sender has not established a prior relationship [5]. The Charity Seals method is unique in that it eliminates the negative aspect of paying to send e-mail by replacing cash payments with charitable donations that can have a positive effect on society.

The next component analyzes DNS and domain records to determine whether the message is likely to have been spoofed or sent from a less reliable SMTP server. SpamGuru's DNS analysis provides most of the advantages of the MARID MTA authentication record [6] without the need for explicit publication of outgoing mail servers. It also can identify new domains and SMTP sessions originating from cable modems and DSL lines. While this information may be insufficient to make a definitive classification, it serves as valuable input to subsequent classifiers.

After establishing the source of a message, we apply an advanced white-listing and black-listing algorithm that determines which items to filter based on user voting. White- and black-lists are maintained both for individuals and for the entire organization. If a user votes a sender as a spammer, the user will not receive messages from that sender again. Other recipients will continue to receive e-mail from this suspected spammer until enough users vote the sender as a spammer to justify global black-listing. By default, SpamGuru is configured to rarely take the recommendations of its white-list and black-list system on face value. Instead, they are used as heavily weighted features by the downstream classifiers. This helps avoid filtering the occasional good message from otherwise spammy senders and vice versa.

At this point, the message content is ready to be analyzed by various message-content filters. First, however, it is parsed by the Intelligent Renderer, which is designed to counter a trick frequently used by spammers to elude anti-spam filters: inserting phony text that is either invisible or likely to be ignored by human readers. To counter these tricks, SpamGuru renders each message to determine what the human user is really likely to see. Rendering is not always the best policy, because it removes from the message the tell-tale features that often indicates spammers' attempts to obfuscate the message, which are very accurate predictors of spam. Therefore, the Intelligent Renderer sends both a rendered and a raw version of the message on through the series of message-content filters.

Each message-content filter has its own tokenization and feature extraction algorithms, which can be critical to the overall success of the filter. In the present setup, the first message-content filter is based on the Winnowing algorithm for plagiarism detection [7]. This algorithm extracts common k-grams appearing in spam and uses these as signatures for identifying nearly identical e-mail. Incoming e-mail that is deemed nearly identical to previously voted spam is also classified as spam. All other e-mail is classified as good. The plagiarism classifier achieves good results with low false-positive rates because of its reliance on near matching. It also can often detect new types of spam before JClassifier or SwiftFile receive enough training examples to learn the new pattern.

JClassifier is a naïve-bayesian text classifier loosely based on Paul Graham's original design [3][4]. JClassifier's additions include a more robust evaluation function, ageing of terms to reduce index size, and a disk-based implementation that is designed to scale to very large mail streams. After JClassifier, SpamGuru applies a regularized linear classifier that has proven to be a top performer in many domains [8]. However, out of the box, this classifier suffers from a high false positive rate. This may improve as we learn to tune the algorithm's runtime parameters. The algorithm is still valuable because its high spam detection rate combines well with other classifiers, as will be shown. Next, we apply a second naive-bayes algorithm that uses the naïve-bayes version of SwiftFile [9]. The SwiftFile classifier has several subtle differences from the Paul Graham formulation, a good summary of which can be found in Sahami, et al. [10].

The final algorithm, and potentially the most interesting, is Chung-Kwei [11]. Chung-Kwei is based on the Teiresias pattern discovery algorithm [12], which has been used successfully to tackle a wide variety of computational biology problems including gene finding and protein annotation [13]. Chung-Kwei uses Teiresias to identify character sequences that appear frequently in spam, but never in non-spam. The original implementation took four days on a 48-node supercomputer to analyze and classify the 173,000 messages used in the experiments below. We have since adapted the original implementation and hand-tuned the algorithms for efficient spam detection. The new implementation processes the same 173,000 messages in 2 hours on a 2.2Ghz Pentium 4 computer. We anticipate further improvements in Chung-Kwei's efficiency and expect it will be practical for real-world deployment in the near future.

4 Evaluation

Ideally, spam filters should be evaluated on large, publicly available databases of known spam and known good mail. Mainly due to privacy concerns, good public databases of both good mail and spam mail do not exist. Instead, many researchers rely on user studies. Unfortunately, there is a strong tendency for users to agree with a classifier's decisions, especially on borderline messages to which the user is indifferent. Furthermore, false positive rates are often grossly underestimated because users never see good mail that is either deleted or placed automatically in a

rarely-inspected Junk Mail folder. Thus user studies often overestimate a system's spam detection rate and underestimate its false positive rate.

We have taken what we believe to be a more accurate and consistent approach to spam filter evaluation. We collected all the messages passing through our SpamGuru pilot server for several days, along with all the user votes and the classifications generated during the same period. Messages not receiving votes were assigned an initial label based on the classifier score received during live operation. This labeling provides a good first approximation, because errors in the initial labeling would be corrected by users by voting. Then, we cleaned the dataset using a combination of automated tools and hand analysis to correct all mistakes we could find in the initial labeling. We used a strict criterion: messages were labeled as spam only if they were unsolicited commercial e-mail. Thus opt-in-spam, in which the user registered at a web site that states that the registration will be used by the web site's partners to send commercial advertisements, is not counted as spam even though many people might still find it objectionable or worthless. The resulting corpus consisted of 173,000 consistently-labeled messages from 200 e-mail users within IBM, roughly 130,000 of which were labeled as spam.

We randomly split the data into equal sized training and test sets, to which each algorithm was applied. Note that, as a consequence of our strict definition of spam, one would expect our measured false positive rates to be a good deal higher than what is typically reported in the literature or in commercial anti-spam product literature. Thus, while our results are internally inconsistent, they are unfortunately difficult to compare with results reported elsewhere.

4.1 Individual Classifiers

Fig. 3 shows the results of evaluating each of our classifiers on the test corpus. Each classifier produces an output score, and classifies a message as spam if its score exceeds a given threshold. The threshold can be treated as an adjustable parameter that governs the tradeoff between the spam detection and false positive rates. For each algorithm, we swept through a broad range of thresholds, measuring the detection and false positive rates at each value. This generated the curves of Fig. 3.

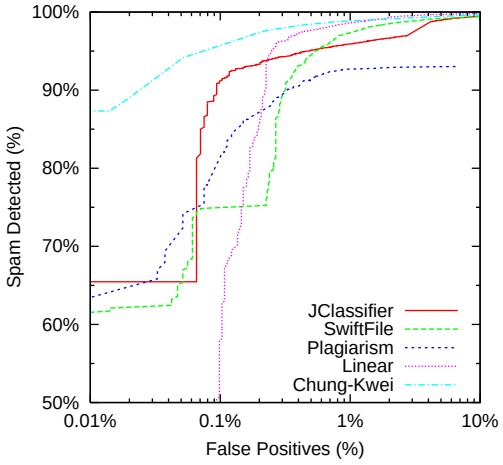

Fig. 3: Individual classifier performance.

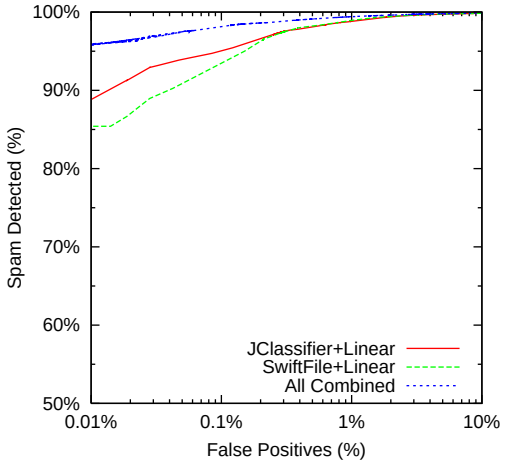

Fig. 4: Combined classifier performance.

Several conclusions can be drawn. First, note that the ideal classifier would have a detection rate of 100% and a false positive rate of 0%, and would hence be represented as a point at the top left corner of Fig. 3. Thus the quality of a classifier can be judged according to how close its curve is to the top left. Chung-Kwei is easily seen to be the strongest classifier throughout the range, especially at low false positive rates. At a false positive rate f of 0.1%, its detection rate d is 95.5%, which exceeds the value of d=91.3% attained by its nearest competitor, JClassifier. Chung-Kwei's superiority is even more evident at lower false positive rates. Chung-Kwei attains d=87.3% at f=0.01%, while JClassifier cannot even register a false positive rate this low. At f=0.1%, the other classifiers have detection rates measuring approximately d=60% at best. Overall, the classifiers have fairly distinct detection vs. false positive curves, with Linear Discriminant doing very well at high false positive rates, and Plagiarism shining at

lower false positive rates. The Linear algorithm fares poorly because its training assumes an equal weighting between false positives and false negatives, so it is likely that cost-sensitive training would improve its performance for small values of *f*.

As has been noted, our strict definition of spam makes it difficult to compare our results directly with those reported elsewhere. As a result, the perceived false positive rate of this algorithm is substantially less than what the absolute performance results suggest. However, JClassifier provides a reasonable calibration point, as it is essentially an enhanced naïve-bayesian classifier that is similar to many spam filters in use today. This suggests that Chung-Kwei is a truly superior spam classifier, although some actual head-to-head comparisons would be required to establish this for certain. A close analysis of the false positives produced by the classifiers at f=0.1% and below revealed that most fell under the category of opt-in-spam, which would have been regarded as spam by most classifiers. Thus the false positives in question are almost universally harmless.

4.2 Combined Classifiers

One of the claimed advantages of our filter pipeline is that we can flexibly combine outputs from multiple classifiers to obtain a combined classifier of superior effectiveness and robustness. To validate this claim, we experimented with many different approaches to combining classifiers. By experimentation, we found that the most successful way to combine classifiers was to use their unthresholded output scores as input to a linear super-classifier. In other words, the super-classifier's score was a weighted sum of the scores of the constituent classifiers, which was then thresholded. The optimal values of the weights and the threshold were established by minimizing a penalty function in which each false negative received a penalty of 1 and each false positive received an adjustable penalty. We generated the curves of Fig. 4 by sweeping through different values of the false positive penalty, using a standard optimization routine to find the best weights and thresholds at each value.

First, we tried all six possible pairings of the four classifiers other than Chung-Kwei, and then we combined all 5 classifiers. The results for two of the six pairs and for the five-way combination are shown in Fig. 4. We find that combining just two classifiers can be very beneficial. Out of the six pairs, the best is JClassifier plus Linear Discriminant. Remarkably, even though both algorithms have the worst possible detection rate of d=0% at f=0.01%, their combination achieves an excellent d=88.6% at f=0.01%, which is slightly better than Chung-Kwei. At f=0.1%, this combination achieves d=95.1%, which is essentially equivalent to Chung-Kwei. The super-classifier places the greatest weight on Linear above f=0.05%, shifting gradually to greater weight on JClassifier at lower values of *f*, although Linear maintains a sizeable weight down to the lowest measurable values of *f*. The other pair shown, SwiftFile plus Linear Discriminant, is also vastly superior to its constituent classifiers at low false positive rates, and the relative weighting of the two constituent classifiers follows the same pattern. Even Chung-Kwei can benefit from combination with other classifiers. The 5-way super-classifier is noticeably superior to Chung-Kwei, attaining d=95.8% at f=0.01% and d=98.1% at f=0.1%. Interestingly, the weight placed on Chung-Kwei in the 5-way super-classifier is in the same general range as that of the other classifiers.

5 Conclusions

We have presented SpamGuru, an anti-spam filtering system for enterprises that is based on three important design principles. First, SpamGuru relieves the burden of anti-spam administration by automating several tasks such as maintaining white- and black-lists, updating filters automatically in response to user votes, etc. Second, the SpamGuru architecture supports easy, flexible configuration. This is important, because one size does not fit all, and because rapid changes in spammer techniques can necessitate changes in configurations or tuning parameters. SpamGuru gives individual users control of their level of filtering (thus accommodating a diverse user base while minimizing complaints), and provides personalized filtering that is usefully combined with global filters based on collaborative voting among users. The filter archive handles users concerns of false positives without the need to call support. Finally, by combining multiple disparate classifiers, we have shown that SpamGuru can achieve excellent discrimination between spam and legitimate mail, and can offer a tunable tradeoff between spam detection rates and false positives, with excellent spam detection even at very low false positive rates.

Acknowledgements

We thank the many researchers who have contributed to the development of SpamGuru, including Nathaniel Borenstein, Vinny Cina, Robert Filepp, Mike Halliday, Shlomo Hershkop, Tien Huynh, Joel Ossher, V. T Rajan, Isidore Rigoutsos, Mark Wegman, and Mary Ellen Zurko. This project would not be possible if not for the daily flood of anonymous e-mail urging us to continue our efforts.

References

[1] B. Leiba and N. Borenstein. A Multi-Faceted Approach to Spam Prevention, *Proceedings of the First Conference on E-mail and Anti-Spam*, 2004.

[2] M. Leonard, M. Rodriguez, R. Segal, and R. Shoop. Managing Customer Opt-Outs in a Complex Global Environment. *Proceedings of the First Conference on E-mail and Anti-Spam*, 2004.

[3] P. Graham, A Plan for Spam. *http://paulgraham.com/spam.html*, August 2002.

[4] P. Graham, Better Bayesian Filtering. *http://paulgraham.com/better.html*, January 2003.

[5] P. Capek, B. Leiba, and M. N. Wegman. Charity Begins at your Mail Server, *http://www.research.ibm.com/people/w/wegman/charity.htm*, 2004.

[6] J. Lyon and M. Wong. MTA Authentication Records in DNS, *Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-marid-core-01.txt*, June 2004

[7] S. Schleimer, D. Wilkerson, and A. Aiken, Winnowing: local algorithms for document fingerprinting. In *Proceedings of SIGMOD 2003*, San Diego CA, June 9-12, 2003.

[8] T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5-31, 2001.

[9] R. Segal and J. Kephart. MailCat: An Intelligent Assistant for Organizing E-Mail. In *Proceedings of the Third International Conference on Autonomous Agents*, May 1999.

[10] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A Bayesian Approach to Filtering Junk E-Mail. *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[11] I. Rigoutsos and T. Huynh. Chung-Kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (SPAM). *Proceedings of the First Conference on E-mail and Anti-Spam*, 2004.

[12] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics*, 14, 55-67, 1988.

[13] I. Rigoutsos, I., A. Floratos, L. Parida, Y. Gao and D. Platt. The Emergence of Pattern Discovery Techniques in Computational Biology. *Metabolic Engineering*, 2,159-177, 2000.

[14] P. Pantel and D. Lin. SpamCop -- A Spam Classification & Organization Program. *In Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[15] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos and P. Stamatopoulos, Stacking Classifiers for Anti-Spam Filtering of E-Mail, In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, 2001.