Open access • Proceedings Article • DOI:10.1145/1811039.1811090

A case for unsupervised-learning-based spam filtering — Source link

Feng Qian, Abhinav Pathak, Yu Charlie Hu, Zhuoqing Morley Mao ...+1 more authors

Institutions: University of Michigan, Purdue University, Microsoft

Published on: 14 Jun 2010 - Measurement and Modeling of Computer Systems

Topics: Bag-of-words model, Unsupervised learning, Supervised learning and Naive Bayes classifier

A Case for Unsupervised-Learning-based Spam Filtering

Feng Qian[1] Abhinav Pathak[2] Y. Charlie Hu[2] Z. Morley Mao[1] Yinglian Xie[3]
[1]University of Michigan [2]Purdue University [3]Microsoft Research

ABSTRACT

Spam filtering has traditionally relied on extracting spam signatures via supervised learning, *i.e.,* using emails explicitly manually labeled as spam or ham. Such supervised learning is labor-intensive and costly, more importantly cannot adapt to new spamming behavior quickly enough. The fundamental reason for needing labeled training corpus is that the learning, *e.g.,* the process of extracting signatures, is carried out by examining individual emails. In this paper, we study the feasibility of unsupervised learning-based spam filtering that can more effectively identify new spamming behavior. Our study is motivated by three key observations of today's Internet spam: (1) the vast majority of emails are spam, (2) a spam email should always belong to some campaign, (3) spam from the same campaign are generated from some template that obfuscates some parts of the spam, *e.g.,* sensitive terms, leaving other parts unchanged.

We present the design of an *online, unsupervised spam learning and detection scheme*. The key component of our scheme is a novel text-mining-based campaign identification framework that clusters spam into campaigns and extracts the invariant textual fragments from spam as campaign signatures. While the individual terms in the invariant fragments can also appear in ham, the key insight behind our unsupervised scheme is that our learning algorithm is effective in extracting co-occurrences of terms that are generated by campaign templates and rarely appear in ham. Using large traces containing about 2 million emails from three sources, we show our unsupervised scheme alone achieves a false negative ratio of 3.5% and a false positive ratio of at most 0.4%. These detection accuracies are comparable to those of the *de-facto* supervised-learning-based filtering systems such as SpamAssassin (SA), suggesting that unsupervised spam filtering holds high promise in battling today's Internet spam.

1. INTRODUCTION

Ever since the onset of Internet spam, supervised machine learning has been the norm in traditional spam filter systems. In supervised learning, in the training phase, a set of labeled training data of spam and ham are fed into a classifier that uses the training corpus to build a model for the spam (and ham). Then in the detection phase, the model is used to classify unknown emails into spam or ham. The difference among various learning algorithms [51] largely lies in the different ways they represent the trained model.

Despite their effectiveness, supervised learning critically relies on the training corpus. Preparing the training corpus is labor-intensive and costly, and potentially leads to a delay in the learning and spam detection cycle. More importantly, any mislabeling of the training corpus (*e.g.,* due to spammers' obfuscation) can affect the effectiveness of the learning process and consequently the spam detection accuracy.

We examine the fundamental reason why these supervised-learning-based schemes require training data. We observe that regardless of the algorithm details, these supervised learning schemes share a common mechanism: learning is performed on *an individual email basis*. In other words, by their very nature of examining one email at a time, these classifiers cannot tell whether an email is spam or ham. This dictates that the only way the classifier knows what information to learn from one particular email to enhance the model is to be explicitly told what the email is.

In this work, we revisit the fundamental reason for needing training data in supervised-learning-based schemes and study the feasibility of a completely unsupervised-learning-based spam filtering scheme. Our study is motivated by three key observations of the spam in today's Internet:

- The vast majority of emails are spam [3, 13, 41];
- A spam email should always belong to some campaign [2, 5, 19, 30, 53, 31];
- The spam from the same campaign are generated from templates that obfuscate some parts of the spam, *e.g.,* sensitive terms, leaving the other parts unmodified [19, 22, 23, 39, 40, 31].

These observations strongly suggest that in principle it is feasible to develop an unsupervised learning scheme that automatically identifies the common terms (signatures) shared by spam belonging to the same campaign and in doing so identifies the spam campaign and extracts the campaign signatures. The signatures can then be used to filter future spam belonging to the same campaign.

While promising, designing such an unsupervised spam filtering scheme faces several fundamental challenges. (1) It requires a robust learning algorithm that can accurately identify common terms shared by spam belonging to the same campaign and consequently the corresponding campaign. (2) The learning algorithm needs to be very efficient so it can be performed online in order to quickly capture campaigns at their onset. (3) Legitimate emails that exhibit similarities such as newsletters and broadcast announcements may be falsely classified into spam campaigns and contribute to false positives in spam detection. (4) The filtering scheme is designed to be deployed in a single organization (or mail service provider), which may not witness enough volume or concentration of spam belonging to a campaign for the mining algorithm to detect the campaign (or soon enough). This can contribute to false negatives in spam detection. There have been a number of previous attempts at unsupervised-learning-based spam detection (*e.g.,* [50, 28, 27, 44]). However, these approaches are not explicitly designed to identify campaigns and are unable to sufficiently expose the similarities (low entropy) shared among the same campaign and consequently suffer from either high false positives or high false negatives.

In this paper, we present the design and evaluation of an online, unsupervised spam learning and detection scheme, SpamCampaignAssassin (SCA), that overcomes the above challenges. SCA performs accurate online campaign detection, extracts campaign signatures, and performs spam filtering using the signatures. To our knowledge, SCA is the first unsupervised spam filtering scheme that explicitly exploits online spam campaign identification and achieves accuracy comparable to the *de-facto* supervised spam filters.

To overcome the first challenge, SCA employs a text-mining framework enhanced by Latent Semantic Analysis (LSA) [24] to cluster spam into campaigns. While LSA has been used to perform spam detection previously [11], building a complete online framework on top of it to perform *campaign detection* is our novel contribution. In SCA, LSA is used to extract the semantics of emails and infer their relationship by producing a set of compact concepts, in which the semantic significance of invariant texts is boosted. To overcome the second challenge, SCA adopts a novel "split-and-merge" strategy in applying the text-mining framework to the incoming email stream to bound the time and space complexity of LSA which in turn enables online spam learning and detection. In particular, the incoming email stream is separated into segments which are processed independently by LSA. The resulting clusters from each segment are then efficiently merged. We show such an approach incurs little loss in learning accuracy. To overcome the third challenge, SCA conservatively performs filtering based on emails' origins and keyword whitelists to effectively reduce the false positive rate.

The fourth challenge, *i.e.,* a possible lack of sufficient spam observed belonging to the same campaign at a single organization, poses perhaps the most significant challenge to unsupervised learning.[1] As shown in our evaluation, our proposed text-mining of email bodies alone generates campaign signatures that cover over 80% of the spam. To overcome such "visibility challenge", SCA employs two optimization techniques. First, it complements the above text-mining-based learning with mining HTML and URL information in email bodies to generate additional campaign signatures. Second, during the online learning and detection process, it self-maintains a source IP blacklist of email sending hosts with a high spam to ham ratio, and it performs lookups of the remaining emails against such a blacklist. Our evaluation results show both techniques effectively encompass the vast majority of the remaining spam.

We have conducted a detailed evaluation of SCA using large traces containing about 2 million emails (over two months) obtained from two heterogeneous organizations (a large university department and a large mail service provider sampled via an open relay sinkhole) and a public email dataset [1]. Our comprehensive evaluation validates campaign identification accuracy, campaign signature quality, and online spam detection accuracy. The results indicate that SCA, as a standalone unsupervised spam detection system, can already achieve spam detection accuracies comparable to those of the *de-facto* supervised-learning-based schemes such as SpamAssassin. In particular, it achieves false negative rates of no more than 3.5%, and false positive rates of at most 0.4%, on the three datasets.

We further manually examined the spam belonging to the 3.5% false negatives and confirmed the reason they escaped SCA is because SCA did not witness enough of similar spam to identify the campaigns they belong to. We show that supplementing SCA with commonly used DNSBL lookup can further reduce the remaining false negative ratio. Ultimately, the visibility problem can be overcome by having multiple domains share their email corpus. We discuss ways of sharing the email corpus anonymously.

In the rest of the paper, we briefly review the traditional, supervised-learning-based spam filter systems in §2, and motivate our unsupervised-learning-based spam filtering design with key observations of today's Internet spam in §3. We present the design of SCA in §4, and evaluate its detection accuracy in offline spam detection in §6 and online spam detection in §7. Finally, we discuss further enhancements to SCA in §8 and related work in §9, and then conclude in §10.

2. SUPERVISED LEARNING IN TRADITIONAL SPAM FILTERING SYSTEMS

In this section, we give a brief overview of supervised learning schemes in traditional spam filter sys-

[1]We note that the same challenge is faced by supervised learning.

tems. In supervised machine learning, spam filtering is recast as a text classification problem using a set of labeled training data. In the training phase, the classifier uses the training corpus to build a model, or a knowledge base. Then in the detection phase, the model is used to classify unknown emails. The difference among various learning algorithms lies in the different ways they use to represent the model. For example, a Naïve Bayes Classifier [26] finds keywords (or other features) that prevail among spam (ham), but seldomly occur in ham (spam), and use them as indicators of future spam (ham). Memory-based learning systems [43] directly store spam or ham instances in the model, then the label of an incoming email is majority-voted by examining the labels of similar emails stored in the model. A support vector machine (SVM) [10] transforms training email documents to data points in a high-dimensional space. Then SVM separates spam and ham by a maximum-margin hyperplane (a hyperplane with the largest distance to the nearest data points in both classes). Other well-known supervised learning paradigms include neural networks [8], maximum entropy models [51], and RuleFit (used by the SNARE system [16]).

Regardless of the algorithm details, these supervised learning schemes share a common ground: learning, *i.e.,* spam signature extraction, is performed on *an individual email basis.* This turns out to be the fundamental reason for the necessity of training data: since the classifier cannot tell whether an email is spam or ham, the only way it knows what information to learn from that particular email is to be explicitly told what the email is. For example, in Bayesian learning, the training data need to be explicitly labeled so the classifier can learn a prevailing keyword is indicative of spam, or ham, or neither.

Supervised learning critically relies on the training corpus to provide useful information, but preparing the training corpus is labor-intensive and costly. More importantly, since supervised learning heavily depends on high-quality training data, feeding inaccurate training corpus (*e.g.,* due to spammers' obfuscation) can lower the detection accuracy, and potentially subvert the entire learning system regardless of how smart the learning algorithm is. These reasons motivate researchers to seek unsupervised spam detection approaches which do not require any labeled training procedure and can automatically and more timely track evolutions of spam campaigns.

3. SPAM CAMPAIGNS AND UNSUPERVISED LEARNING

In this section, we motivate our unsupervised-learning-based spam filtering with three key observations of today's Internet spam: (1) the vast majority of emails are spam, (2) a spam email should always belong to some campaign, (3) spam messages from the same campaign are typically generated from some templates that obfuscate some parts of the spam, *e.g.,* sensitive words, but not all of it. The common (unobfuscated) parts of the spam belonging to the same campaign effectively lower the "entropy" of these spam and lend themselves to unsupervised learning.

3.1 Spam Campaigns

Numerous recent spam reports have suggested that the vast majority of emails in the Internet today are spam. A report in May 2009 by Symantec suggests that 90.4% of emails were spam [41], Google (postini) rated spam volume to be around 90-95% in all four quarters of 2009 [13], while a report from Microsoft in September 2009 rated spam to be at 97.3% of emails [3]. Several reports [18] estimate about 85% of spam originate from botnets. Symantec [41] estimates that 56.7% of spam originate from known botnets, while a study [19] estimates 79% of spam messages arriving at University of Washington campus originated from just 6 botnets.

Every spam has an objective, for example, to sell a product from a particular vendor, to advertise a political campaign [33], or to infect hosts to recruit them into a botnet. Spam messages with the same objective belong to a spam campaign (SC). To be effective, a spammer generates spam in units of campaigns, each consisting of a large volume of spam messages sharing the same objective and targeted at thousands to millions of mailboxes of many destination domains. Therefore, due to the high volume of spam campaigns, a significant number of the spam in a SC are expected to be received at an individual end destination domain.

Recent studies on offline spam campaign analysis [2, 5, 19, 30, 53] confirm that the majority of spam received at an end domain/spam honeypot can be clustered into SCs. John *et al.* [19] estimated that 89.2% of emails to UW were spam and more than 95% of spam feed contained URLs. 80% of the spam pointed to just 11 distinct web pages whose content did not change during their period of study (50 days), suggesting that at least these 80% of spam belonged to SCs (at most 11 SC selling 11 products). Pathak *et al.* [30] manually estimated that, in a spam-trace collected at a relay-sinkhole, 58.6% of spam contained URLs, of which 65% belonged to just 7 SCs, each containing thousands of spam. Out of the remaining 41.4% spam that did not contain a URL, 76% were identified to be a part of some SC, using features like unique phone numbers, Skype ids, and mail ids in spam messages. The remaining 24% of spam that did not contain URLs were believed to be part of a pump-dump stock SC. Calais *et al.* used Spam Miner [6, 5] to cluster 97.5 million spam obtained during a period of one year at 10 honeypots spread over 5 broadband networks into 16K SCs. Over 99.999% of the SCs consisted of more than 1K spam messages each.

3.2 Template-based Campaign Generation

Recent studies on spam campaigns further uncovered ample evidence that spam in a SC are typically generated from spam templates [19, 22, 23, 39, 31]. A spam template defines a skeleton of spam, along with a set of rules from which many varying spam can be generated. The rules specify how the headers should look like, which part of text/body should be obfuscated, where in the body the URLs should be placed, which words should be rotated, etc. Spam templates are dispatched to bots which use them to generate individual instances of spam and send one to each recipient. The bots/spammers are known to use tools, *e.g.,* DarkMailer [9], send-safe, and the reactor-mailer tool [39], to automatically generate spam based on templates.

When designing a template, spammers try hard to ensure that the spam generated from the template is unlikely to be flagged by popular anti-spam solutions. To do so, spammers refine the templates to obfuscate and rotate key words, *i.e.,* words that are sensitive in identifying a message being a spam email. Some of these tools even come integrated with open-source anti-spam solutions to self-check the scores of spam generated from an input template [39]. While emphasis is placed on key words, words surrounding the obfuscated text often receive less attention by both spammers and regular-expression-based body tests in spam filters. To illustrate this point, let us consider a spam template from dark-mailer [9].

```
{%Canadian|CANADIAN%}
{%drugs|meds|health treatments|health remedy's|
treatments|health drugs%}...the{%only|simply|merely|
purely|easily|straightforward|clearly|obviously%}
{%way|path|route|approach|method|mode%} to {%go|buy|
acquire|get|purchase%}

{%URL_ADDRESS%}

"Ah, {%made|dust|board|scorch|receipt|tail|commercial|
debt|comparison%}{%strap|language|trade|cart|
thundering|stuck|inquisitively|tactic|hand%} your
{%grip|plate|inquisitive|steer|splendid|promptly|
needle|shoe|drink%} excellency, I am{%ball|post|
opinion|library|lighten|cow|letter|weak%} overwhelmed
with deligh "Yes."
```

The above template describes just the body part of a complete template. Word rotation is being used to obfuscate the whole body text. One word is picked up from every braces - {%..%}. Even from such a simple template, millions of distinct spam can be generated. While all the sensitive words are rotated in the spam, the word sequences "*...the*", "*excellency, I am*", "*overwhelmed with deligh "Yes."*" will co-occur in every spam generated from the template.

Finally, the templates for a particular SC are not frequently changed. A study involving the storm botnet [23] observed that half of the campaigns from the storm botnet employ only a single template. Longer running campaigns employ more templates. In summary, template-based spam generation employed by modern spammers suggests that spam belonging to the same campaign are highly likely to share a set of common terms in the body.

3.3 Towards Unsupervised Learning

The above observations about the spam in today's Internet suggest that in principle it is feasible and highly promising to develop an unsupervised learning scheme that automatically identifies the common terms shared by spam belonging to the same campaign and in doing so identifies the spam campaign and extracts the campaign signatures, to be used for filtering future spam of the same campaign.

As discussed in §2, a fundamental reason that supervised learning needs labeled data is that when examining emails one at a time, it is difficult to distinguish spam from ham. However, when examined together, the collective common properties of spam at the campaign level gives a clear manifestation of themselves:

*Statistically, due to the nature of template-generated spam campaigns, spam bear lower entropy (*i.e., *higher similarities) than legitimate emails.*

In a spam campaign, the spammer, *e.g.,* using a botnet, sends a large number of spam messages, of the same intent, in an automated fashion. In contrast, legitimate emails are manually sent out by human for different purposes. Such a fundamental disparity is inevitably reflected on the entropy (*i.e.,* dissimilarity) of the email contents; spam belonging to a campaign tend to exhibit low entropy, and ham tend to exhibit high entropy. An unsupervised learning algorithm should be able to uncover the entropy gap between spam and ham, by accurately discovering frequent invariant textual parts among the spam in a campaign.

While spammers constantly attempt to increase the entropy of spam by escalating their obfuscation techniques, *e.g.,* polymorphic spam instances [39], we argue that the entropy gap between spam and ham will never disappear due to the fundamental disparity between campaign-based spam and legitimate emails which are personalized in nature[2]. In particular, even if all the sensitive keywords (*e.g., Viagra* and *Replica*) are obfuscated at a per-spam basis, it is very likely that there exists *co-occurrence* of non-sensitive words in the templates that are not obfuscated and unlikely to co-occur in legitimate emails. Otherwise, spammers have to spend considerable amount of efforts and resources in customizing each spam instance.

While promising, the design of such an unsupervised

[2]While in principle a spammer could use obfuscation/rotation for all terms in the template, doing so would severely affect the readability of spam messages.

spam filtering scheme faces several fundamental challenges. (1) It requires a robust learning algorithm that can accurately identify common terms shared by spam belonging to the same campaign and extract the campaign signatures. (2) The learning algorithm needs to be very efficient so it can be performed online in order to quickly capture campaigns at their onset. (3) Legitimate emails that exhibit similarities such as newsletters and broadcast announcements may be falsely classified into spam campaigns and contribute to the false positive ratio in spam detection. (4) Same as any practical spam filtering solution, such a scheme should be designed to be operated in a single organization. Hence it may not witness enough spam belonging to a campaign, *e.g.,* in a short interval after the onset of that campaign, for the mining algorithm to detect the campaign. This can happen as a campaign may not generate the same amount of concentrated spam to all organizations or domains. This potential "visibility" limitation can contribute to the false negative ratio in spam detection.

3.4 Previous Unsupervised Approaches

There have been a number of previous attempts at unsupervised spam detection. To the best of our knowledge, all of these work were motivated by the same set of observations as ours, without perhaps explicitly stating them. However, we found none of the existing approaches are capable of sufficiently exposing the similarities (low entropy) shared among the same campaign and consequently they all suffer from either high false positives or high false negatives.

The simplest unsupervised approach is hash-based duplicate detection. This scheme, however, can be trivially subverted by introducing the slightest obfuscation. The authors in [50] employ a similar idea except that they use a hash vector (the hash values of the first N k-grams) instead of a single hash. Although the authors claim they achieve 98% recall rate and 100% precision rate, their evaluation method is questionable as they inject identical copies of the same "pseudo" emails. The approach in [28] is based on the assumption that the distribution of substrings in spam (campaigns) is different from Zipf's law which legitimate documents exhibit, due to the large amount of similarities shared by the spam. The authors in [27] propose a more complicated method called String Alienness which detects spam by discovering outliers in the substring frequency distribution. However, these approaches suffer from poor accuracy. The evaluation in [27] shows that the highest precision and recall rate range from 72% to 80% and 63% to 89% in four datasets, respectively[3]. In [44], the authors employ an entropy-like measure, called Document Complexity, whose values are high for ham, but low for spam generated from "seed documents" (*i.e.,* campaign templates). However, their evaluations using four datasets containing a total number of 6K blog spam shows that the highest precision and recall rate range from 71% to 95% and 63% to 82%, respectively. A recent theoretical work [15] by Haider *et al.* detects spam campaigns using Bayesian clustering that transforms email documents to independent binary feature vectors. Their algorithm works well in supervised mode, but when no training hams are available, its false negative rate is at least 20%. In [47], the authors propose a semi-supervised approach where the system first clusters unlabeled emails, then the user only needs to label each cluster instead of each email instance.

[3]We did not find accuracy evaluation in [28].

4. SPAMCAMPAIGNASSASSIN (SCA): UNSUPERVISED SPAM FILTERING SCHEME

In this section, we present SpamCampaignAssassin (SCA), an online, unsupervised spam learning and detection scheme. SCA is explicitly designed to be campaign-oriented. The key component of SCA is a novel text-mining-based framework that clusters spam into campaigns and extracts the invariant textual fragments from spam as campaign signatures. SCA further complements text-based signatures with two robust algorithms that extract campaign signatures from HTML structures and URLs in the mail body to further reduce the false negatives. A further optimization on online spam detection of SCA is presented in §7.2.

4.1 Text-mining Framework for Extracting Textual Signatures

Figure 1 shows the architecture of our online text-mining framework. Our framework takes as input the incoming email stream which may contain multiple spam campaigns (spam) and legitimate emails (ham), performs unsupervised learning, and outputs textual signatures that well separate spam from ham.

The core unsupervised learning engine in our online text-mining framework performs Latent Semantic Analysis (LSA) [24] and clustering. Since LSA has a quadratic complexity, we adopt a "split-and-merge" approach in our framework. First, the incoming email stream is split into multiple *segments*, each consisting of emails received in a continuous duration. Processing results, *i.e.,* preliminary identified clusters and signatures, for multiple segments are efficiently merged based. Such a "split-and-merge" design has two benefits. First, it enables our framework to process multiple segments in parallel, *e.g.,* on a multicore machine. Second, more importantly, it makes it feasible to apply sophisticated but expensive machine learning algorithms such as LSA [24]. Let the total number of emails be n and the number of segments be s. After segmentation, the cost of a polynomial algorithm of degree k decreases from $\Theta(n^k)$ to $\Theta\left(s \cdot \left(\frac{n}{s}\right)^k\right) = \Theta\left(\frac{n^k}{s^{k-1}}\right)$, by a factor of s^{k-1}, with little loss of accuracy as our evaluation will show. We now

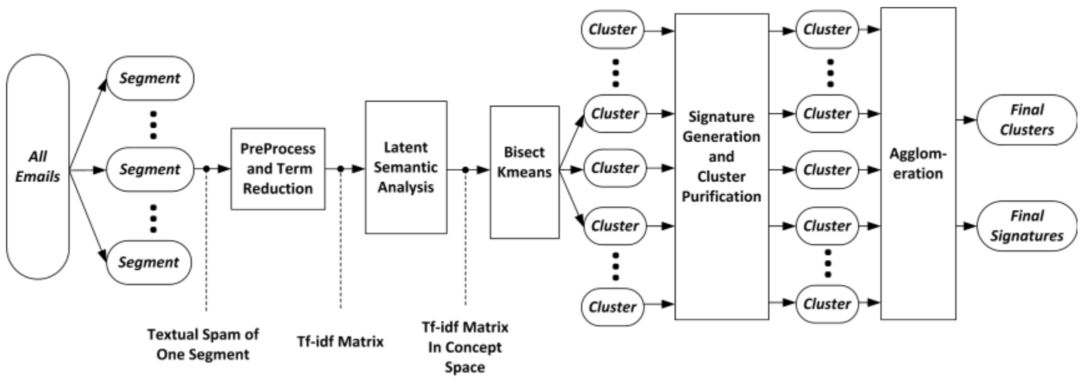

Figure 1: **Text-mining framework for extracting textual signatures.**

describe each framework component in detail.

4.1.1 *Preprocessing: term reduction and matrix generation*

The input to this step is a segment of unlabeled documents. After filtering 965 common English stop words and HTML tags, we perform *term reduction* to mitigate obfuscated terms that are usually random (low-frequency) terms injected by the spammer, and to boost the performance of LSA (§4.1.2) and vector-based clustering (§4.1.3), as their performance degrade as the term space (vocabulary size) becomes larger [37]. In our implementation, first, terms with document frequency (the number of documents in which a term occurs) less than a threshold tf are removed. Second, we employ Porter Stemming Algorithm to reduce English words to their basic forms. [48] shows that it is possible to reduce the vocabulary size by a factor of 10 (100) with no loss (very small loss) of accuracy.

After term reduction, we represent the documents using the common *tf-idf* (term frequency - inverse document frequency) weighting considering both the importance of the term in a particular document as well as the whole corpus of documents. Thus we tokenize the corpus into a *tf-idf* matrix where the value of (i, j) corresponds to the composite weight for term i in document j.

4.1.2 *Boosting semantics of invariant texts: Latent Semantic Analysis (LSA)*

After tokenizing the documents, the *tf-idf* matrix X is usually large in its dimension. In order to boost the semantics of invariant texts, we employ Latent Semantic Analysis (LSA) [24] to transform the original vocabulary space into a compact concept space. For example, assume the vocabulary consists of three terms: {*viagra*, *cialis*, *replica*}. Assume *viagra* and *cialis* have high co-occurrence in the corpus, but the correlation between *replica* and the other two terms is very low. Then the concept space generated by LSA may contain two terms $\{\alpha{\cdot}\textit{viagra} + \beta{\cdot}\textit{cialis}, \textit{replica}\}$ where the values of α and β depend on how well *viagra* and *cialis* are correlated. The first "term" in the concept space can be thought as *drug*.

Mathematically, let the original *tf-idf* matrix be an $m \times n$ matrix X where m is the number of terms and n is the number of documents in the segment. The LSA algorithm transforms X to an $r \times n$ matrix X' by generating a low-rank (dimension) approximation to X based on singular value decomposition (SVD) of X. The low-rank approximation X_r is constructed as $X = U\Sigma V^T \approx U_r\Sigma_r V_r^T = X_r$ where Σ_r contains the r largest singular values of X and the matrix X_r is the closest r-rank approximation to X in terms of the Frobenius norm. Effectively reducing the vocabulary size from m to $r \ll m$ significantly increases the accuracy and performance of the subsequent clustering procedure.

Unfortunately, LSA is not scalable. Its time complexity (the complexity of SVD) is $\Theta(nm^2)$ for an $m \times n$ matrix. Thus, directly applying LSA on a large dataset is computationally infeasible. This was the main motivation for splitting the incoming email stream into segments to bound n and hence limiting m.

4.1.3 *Separating low-entropy emails: clustering*

Given the corpus containing n vectors in the concept space, X_r, the next natural step is to separate its low-entropy components (*i.e.,* spam campaigns). This is achieved by a clustering procedure that, without supervision, finds groups of documents with similar content based on a predefined distance function between two vectors. We employ the bisect k-means clustering algorithm [42] due to its linearity in time and space and its ability to automatically decide the number of clusters. The typical Cosine dissimilarity (one minus the cosine of the included angle between two vectors) is used as the distance measure.

The bisect k-means algorithm works as follows. Initially there exists only one cluster consisting of all documents. The algorithm iteratively picks one cluster with the worst quality, then bisects it into two clusters. The cluster quality of a cluster $C = \{\mathbf{v_1}, ..., \mathbf{v_n}\}$ is defined as $q(C) = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{v_i} - \mathbf{c}|$ where $\mathbf{c}$ is the centroid of C, *i.e.,* $\frac{1}{n}\sum_{i=1}^{n} \mathbf{v_i}$. A smaller $q(C)$ indicates better cluster quality. The algorithm terminates when $\forall$ cluster $C : [q(C) < \delta \vee |C| < \lambda]$ where δ and λ are predefined thresholds for the cluster quality and cluster size, respectively.

4.1.4 Preliminary signature generation: computing frequent term sequences

We exploit the similarities among the vectors in the same cluster to generate preliminary signatures, which are invariant textual fragments prevailing in the cluster. We compute π-frequent term sequences, which are consecutive term sequences with the document frequency higher than π, as preliminary signatures (we will refine these signatures later). Those π-frequent term sequences could be efficiently obtained by scanning the entire cluster once using the suffix tree algorithm [14]. We prefer consecutive terms sequences to other design choices for textural signatures due to its linear computation cost and its robustness to obfuscation (other approaches we investigated include N-gram terms, non-consecutive terms, term sets, and the BLAST pattern discovery algorithm [14] used in bioinformatics). Again, the robustness of signatures comes from the fact that SCA extracts frequently appearing co-occurrences of terms that are generated by campaign templates but are rarely appearing in ham.

4.1.5 Purifying clusters

The clustering step effectively exposes similarities shared by spam in the same campaigns, as the entropy in each cluster is much lower than the entropy of the entire corpus. However, as clustering is a partition operation, every email in the corpus is classified into a certain cluster. In order to obtain useful, precise signatures, we need to remove the noises (*i.e.,* legitimate emails) in the clusters. We call this procedure *cluster purification*, which consists of four steps:

1. **Filtering legitimate emails.** This step leverages the preliminary signatures generated in the previous step. For each cluster, we remove emails not covered by any preliminary signatures. The majority of these are legitimate emails mingled in the clusters.

2. **Filtering clusters with few origins.** We eliminate the entire cluster if all messages are originated from no more than u distinct IPs where u is a threshold. Such a cluster is more likely to be a legitimate campaign (*e.g.,* newsletters and broadcast announcements) or a long threaded discussion, since the *de facto* spamming strategy today is to exploit botnets involving a large number of infected hosts, as discussed in §3. Also, emails from internal IP addresses of the organization where SCA is deployed are automatically whitelisted.

3. **Eliminating small clusters.** We eliminate the entire cluster if its size (*i.e.,* the number of emails in the cluster) is less than λ, a predefined threshold.

4. **Eliminating legitimate clusters.** Mailing lists may exhibit behaviors very similar to spam campaigns *i.e.,* messages from heterogeneous sources share similar content. We use keyword whitelists (they are test against the signatures) to filter clusters containing legitimate mailing list messages. The whitelist also contains textual fragments that may frequently occur in legitimate mail bodies (*e.g.,* the name of the organization), based on the administrator's domain knowledge. Given that the textual behavior of ham is much more stable than that of spam, we expect the maintenance overhead of whitelists to be low.

At the end of cluster purification, we update the signatures accordingly, *i.e.,* we remove signatures that are no longer π-frequent term sequences in the purified clusters.

4.1.6 Agglomerating similar clusters for all segments

Due to the split-and-merge operational model of SCA, all operations described from §4.1.1 to §4.1.5 are performed on a per-segment basis. It can happen that one actual spam campaign in the same segment may be broken down into multiple clusters (due to the over-partition of bisect k-means) or a campaign persists over multiple segments. The signatures for these clusters due to the same campaign are likely to be very similar. To avoid a large number of redundant signatures, when all segments are processed, our framework agglomerates all purified clusters into final clusters with the purpose of reducing the number of signatures. The agglomeration is essentially the reverse procedure of clustering. Initially, every cluster forms an agglomerated cluster (which is a set of clusters) itself. Then we repeatedly merge two similar agglomerated clusters based on their signature information until no further merge can be performed.

Algorithm 1 Cluster Agglomeration

```
Input: $n$ purified clusters $C_1, ..., C_n$.
Output: $A$, the set of agglomerated clusters.
Parameter(s): Two thresholds $\rho1$ and $\rho2$.
1: $A \leftarrow \{\{C_1\}\} \cup \{\{C_2\}\} \cup ... \cup \{\{C_n\}\}$
2: while $\exists A_i, A_j \in A$ :
     $[\forall C_x \in A_i, C_y \in A_j : \exists \alpha \in C_x, \beta \in C_y : T_{ovl}(\alpha, \beta) > \rho_1] \wedge$
     $[\forall C_x \in A_i, C_y \in A_j : \exists \alpha \in C_x, \beta \in C_y : S_{ovl}(\alpha, \beta) > \rho_2]$
   do
3:     $A \leftarrow (A - \{A_i\} - \{A_j\}) \cup \{A_i \cup A_j\}$
4: end while
```

Our novel cluster agglomeration algorithm is formally described in Algorithm 1. Initially, every cluster C_i forms an agglomerated cluster A_i by itself. Then we repeatedly merge two agglomerated clusters A_i and A_j. A_i and A_j can be merged if they satisfy the two conditions in Line 2 of Algorithm 1:

1. **All cluster pairs share similar signatures.** The similarity of two signatures α and β is measured by $T_{overlap}(\alpha, \beta)$, which computes the length of longest non-consecutive (but ordered) sequence (LCS) shared by α and β.

2. **Shared signatures are informative.** This is measured by $S_{overlap}(\alpha, \beta)$ which computes the sum of *idf* values (inverse document frequency *w.r.t.* the entire dataset) for each term in α and β. A high $S_{overlap}$ value indicates that α and β are "informative" signatures in the sense that they do not prevail among all emails in the dataset, as it is known that the terms with low-to-medium document frequency are the most informative ones [36]. The $S_{overlap}$ measure is especially useful when LCS is short.

4.2 Extracting HTML and URL signatures

HTML and URL are popular elements in today's spam. In our datasets, we observe a non-trivial fraction of spam (especially short spam) have URLs as the main part of the body. We also observe that for many campaigns, their HTML structures are much more stable than the obfuscated texts. These observations motivate us to design two robust (*i.e.,* conservative) algorithms to extract HTML and URL based campaign signatures that are complementary to textual signatures, in order to reduce false negatives in spam detection without increasing false positives.

Unlike the texts in email bodies which can be represented linearly by vectors, HTML contents form nonlinear tree structures. We adopt a simple approach that aggregates HTML trees by exact matching, based on our observation that HTML structures are much less obfuscated than texts, and that most popular spamming tools do not provide HTML-based obfuscation [39]. To further reduce false positives, we ignore trees with fewer than 20 nodes since simple trees are more possibly shared between spam and ham. We could design or make use of more sophisticated algorithms (*e.g.,* tree edit-distance based clustering [29]) to handle HTML obfuscations, but we leave it as our future work.

For URL-based signature extraction, we focus on a particular type of URLs (examples are shown in Table 7 in §6.2) that consist of both the *invariant* and the *obfuscated* components in the URL string, for the following reason. The invariant component indicates the low-entropy nature of spam campaigns, while the obfuscated component well separates spam from ham since it is common that the same URL without obfuscated components occurs in multiple legitimate emails (*e.g.,* newsletters).

Algorithm 2 URL Signature Extraction

```
Input: n URLs $U_1, ..., U_n$ extracted from an email corpus.
Output: $S$, the set of URL signatures
Parameter(s): $\lambda, \varepsilon_{\text{url}}, \varepsilon_{\text{ip}}$
1: $S \leftarrow \lambda$-frequent domain parts
2: for each $s \in S$ do
3:     if URLEntropy($s$) $< \varepsilon_{\text{url}} \vee$ IPEntropy($s$) $< \varepsilon_{\text{ip}}$ then $S \leftarrow S - \{s\}$ endif
4: end for
```

The signature extraction procedure is depicted in Algorithm 2. After preprocessing URLs by only keeping their domain parts, we focus on the invariant component by computing all λ-frequent domain parts [4](recall that in §4.1.5, λ is the threshold for cluster size). Next, we require URLs that are covered by a λ-frequent domain part to exhibit obfuscations, *i.e.,* the URLEntropy is greater than a threshold ε_{url}. We also require emails that are covered by URLs with a λ-frequent domain part to be from "random" sources, *i.e.,* their IPEntropy is greater than a threshold ε_{ip}. Given a list of URLs (only with their domain parts), the URLEntropy is defined as $-\sum_u p(u) \log p(u)$ where $p(u)$ is the frequency of a specific URL u in the list. Similarly, given a list of IPs, we define the IPEntropy as $-\sum_i p(i) \log p(i)$ where $p(i)$ is the frequency of a specific IP i in the list.

To further reduce false positives, we also employ cluster purification (Steps 2 and 3 in §4.1.5) for HTML and URL signatures by regarding all emails covered by each HTML or URL signature as a cluster. As will be shown in §6, for our largest dataset, HTML and URL signatures, when used alone, contribute up to 39.8% to 27.4% true positives, respectively, with less than 0.1% false positives.

5. EVALUATING TEXTUAL CLUSTERS

Since SCA is explicitly designed to be campaign–oriented, *i.e.,* it identifies campaigns and then uses the campaign signatures to perform spam detection, we proceed with our evaluation of SCA in three steps. In this section, we evaluate the accuracy of the generated campaign clusters. We then evaluate the detection accuracy of the three types of signatures in §6. In these two sections, SCA performs *offline* detection where we generate signatures using campaigns agglomerated from clusters in all segments in an unsupervised manner. Then the signatures are applied to the entire dataset. Finally, in §7, we present the design details and evaluation of the *online* version of SCA.

[4]A domain consists of multiple parts separated by ".", and frequent parts at different positions are treated as distinct frequent parts, *e.g.,* `*.foo.*.*` and `*.*.foo.*`.

5.1 Data Sets

Table 1 describes the three data sets used in our evaluation. The DEPT trace was collected from two mail servers serving about 2K users in a large university department. We also have labels classified by SpamAssassin for most emails, but we do not fully trust the results as SpamAssassin may not be 100% accurate. The RELAY trace [30] was a subset of one month's trace collected from an open relay located in Eastern US which has been running since Oct 2007; only emails going to the destination domain of yahoo.com were kept in the subset. We note all emails collected by the open relay are spam since the presence of relay was not announced to anyone and legitimate mails are not sent through random relays. Since there are other sources that spam yahoo.com, the RELAY trace represents a sample of all the spam going to yahoo.com during the collection period. We also use one public dataset, ENRON [1], to evaluate false positives of campaign signatures in §6. The ENRON corpus consists of 0.5M emails (most are ham) collected from a large company. Since the ham are mingled with a small amount of spam, we pick a subset of 37.9K emails from all users' sent-boxes, thus we are confident that emails in the subset are purely ham. We use a short and fixed whitelist (§4.1.5) of 20 terms for DEPT, but no whitelist is applied to RELAY and ENRON.

5.2 Parameter Setting

Table 2 shows the parameters used by SCA, for processing the three datasets[5]. We note that although the datasets have significant disparities in terms of content, spamming sources, and languages, we found that the same set of parameters yield good results for all segments in three datasets.

Among all parameters, three have a large impact on the text-mining performance: n (the segment size), tf (the frequency threshold for term reduction), and r (the dimension of the concept space), because the computational complexity of LSA (the bottleneck operation of the framework) is $\Theta(nm^2 + nr^2)$ where m, the vocabulary size, depends on tf. We empirically tried (for DEPT and RELAY) *(i)* tf = 30, 50, 100, 150, 200 when fixing r =300 and n = 15K; *(ii)* r = 100, 200, 300, 400, 500 when fixing tf = 50 and n = 15K; *(iii)* n = 10K, 15K, 20K, 40K when fixing r = 300 and tf = 50. All configurations show no qualitatively difference in clustering accuracy. This indicates that we could use small values of n, r, tf in online detection for better text-mining performance. In particular, We note that the third experiment that varies segment size n suggests segmenting incoming emails for per-segment text-mining analysis does not worsen the potential visibility problem faced by SCA as mentioned in §3.3.

The rest of the parameters only affect the accuracy.

[5]For the value of ρ_1 in Table 2, l_α and l_β are the lengths of two signatures α and β, respectively.

Table 3: Cluster quality under sampled evaluation.

Dataset	DEPT	RELAY
# Examined clusters	500 largest clu.	100 largest clu.
# Samples per cluster	50	100
# Quality: good	488 (97.6%) clu.	100 (100%) clu.
# Quality: acceptable	10 (2%) clu.	0
# Quality: incorrect	2 (0.4%) clu.	0

Their values are derived from the general knowledge about spam campaigns (π, λ, u), and empirically from the datasets (δ, $\rho_1, \rho_2, \varepsilon_{\text{url}}, \varepsilon_{\text{ip}}$). Again, since they work well on the three diverse datasets, we believe they are fairly general and applicable to other datasets.

5.3 Manually Sampled Evaluation

The framework generates 1,775 and 243 agglomerated clusters on DEPT and RELAY datasets, respectively. Note we are only concerned with the quality of individual clusters; it is acceptable that spam in a campaign are split into multiple clusters (*e.g.,* due to over-partitioning (§4.1.3)) as each cluster will contribute some signatures for spam detection. Automated evaluation of textual clusters is challenging due to lack of ground truth. Instead, we perform sampled manual inspection as follows. For DEPT, we pick the largest 500 clusters and visually sample 50 spam in each cluster. Then we score the cluster quality with one of the following three marks: *good* if all 50 spam unanimously belong to the same campaign due to similar content and text structure; *acceptable* if less than 10 emails are misclassified (they are legitimate emails or they belong to other campaigns); *incorrect* if at least 10 emails are misclassified. For RELAY, we sample 100 emails in each of the largest 100 clusters and use the same evaluation metrics. The evaluation results shown in Table 3 demonstrate the text-mining framework is highly accurate in identifying campaign clusters.

5.4 Manually Unsampled Evaluation

Leveraging the RELAY dataset, we further perform manual unsampled evaluation of the quality of the clusters generated by our text-mining framework. We manually identified five campaigns (SC1 to SC5 in the RELAY dataset, as shown in Table 4. They were used in our previous study [30]). If we feed the mixed dataset containing SC1 to SC5 to the text-mining framework that produces several agglomerated clusters, then ideally two emails from different campaigns in Table 4 should not be mingled in the same cluster. This is validated by the results shown in Table 5 where "misclassified" denotes the number of spam that belong to clusters that contain emails from two sets, and "false negatives" denotes the number of spam not belonging to any cluster. Furthermore, Table 5 shows the clear benefits of the cluster purification step (§4.1.5), which significantly reduces misclassified spam with acceptable tradeoff of false negatives.

Table 1: **Email traces used for evaluation.**

	DEPT	RELAY (dest. to Yahoo)	ENRON [1]
# Emails	1.68 M	316.4 K	37.9 K
Sampling	1:1	1:10	Unknown
Time	April 5 - May 9, 2009	February 1-28, 2009	2004
Source	Dept. mail servers	Open relay	Sent-boxes of 150 employees
Type	Spam + ham	All spam	All Ham
Language	Mostly English	Mostly Chinese	Mostly English
Labels	SA labels	Ground truth	Ground truth
§	§5, §6, §7	§5, §6, §7	§6

Table 2: **Parameters used in the text-mining framework.**

Parameter Name	§	Value
n: segment size	§4.1	15,000
tf: freq. threshold for term reduction	§4.1.1	50
r: concept space dimension	§4.1.2	500
Threshold parameters		
δ: for vector similarity	§4.1.3	0.45 to 0.5
π: for document freq. of textual signatures	§4.1.4	50% to 60%
λ: for cluster size	§4.1.5, §4.2	30
u: for # IPs in legitimate clusters	§4.1.5	10
ρ_1: for T_{ovl} in cluster agglomeration	§4.1.6	$0.8\max\{l_\alpha, l_\beta\}$
ρ_2: for S_{ovl} in cluster agglomeration	§4.1.6	5.0
$\varepsilon_{\text{url}}, \varepsilon_{\text{ip}}$: for URL/IP entropies	§4.2	3.0

6. EVALUATING CAMPAIGN SIGNATURES

We now evaluate the offline spam detection accuracy of the three types of signatures, using the three datasets (DEPT, RELAY, and ENRON). This helps us to understand the upper-bound on the effectiveness of the derived signatures before we build the online version of SCA with additional optimizations.

6.1 Methodology

In §6 and in §7, we use TP, TN, FP, FN to denote true positive(s), true negative(s), false positive(s), and false negative(s), respectively. We employ two widely used metrics for detection accuracy: False Positive Rate (FPR) and True Positive Rate (TPR, or detection rate). FPR is defined as FP/(FP + TN), and TPR is defined as TP/(TP+FN). A perfect detection scheme will have a FPR of 0 and a TPR of 100%.

The evaluation metrics are intuitive, but calculating them is slightly challenging. We apply generated signatures on all emails in the dataset, then measure how many emails are misclassified. For RELAY (ENRON) this is straightforward since all emails are spam (ham). For DEPT, however, since we do not fully trust the SpamAssassin (SA) labels, we perform the evaluation by classifying emails into four categories. *(i)* Both SA and our detection scheme SCA report spam, then we are confident that such emails are indeed spam (TP); *(ii)* Both SA and SCA report ham, then we assume that they are truly ham (TN); *(iii)* SA reports spam, but SCA reports ham, then we conservatively regards such emails as spam (*i.e.,* we assume SA has no FP), therefore they are FN of our detection scheme; *(iv)* SA reports ham, but SCA reports spam, then we manually sample 500 emails to find out how many emails are misclassified by SpamAssassin therefore they are our TP (we do assume SA has FN), and how many emails are misclassified by us *i.e.,* they belong to our FP. Let the number of emails in Category *(i)*, *(ii)*, *(iii)*, and *(iv)* be n_1, n_2, n_3, n_4 and the fraction of emails misclassified by SA in Category *(iv)* be r, $0 \leq r \leq 1$, based on our sampled manual inspection on emails in Category *(iv)*. Then FP $= n_4(1-r)$ and TP $= n_1 + n_4 \cdot r$ and we can compute FPR and TPR as FPR $= n_4(1-r)/[n_4(1-r)+n_2]$, and TPR $= (n_1 + n_4 \cdot r)/(n_1 + n_4 \cdot r + n_3)$.

6.2 HTML and URL Signatures

For the DEPT trace, SCA generates 679 HTML signatures and 1647 URL signatures. Their coverage (the number of spam that are covered by each signature, sorted in descending order) is plotted in Figure 2 where both types of signatures exhibit heavy-tail distribution, *i.e.,* a small number of signatures can be used to detect a large fraction of spam. The most effective HTML and URL signatures capture 55K and 88K spam, respectively, out of the total 1.68 million emails. Two example URL signatures and their covered obfuscated URLs are listed in Table 7. Table 6 shows the spam detection accuracy when using the three types of signatures individually and when using them together. As shown in Columns 2 to 4, HTML and URL signatures incur very low false positives (less than 0.1%), but their detection capabilities are limited. Even by combining both, only 54.6% of spam are detected (Column 4 in Table 6).

Set (Size)	Category	How they form a campaign
SC1 (11K)	Drugs, adult video	Contain a distinct URL
SC2 (62K)	House renting	Contain a distinct URL
SC3 (36K)	Books and videos	Contain similar URLs
SC4 (55K)	Financial services	Contain similar URLs
SC5 (33K)	Mortgage loan service	Contain a unique Skype ID

Table 4: Five manually identified campaigns used in unsampled evaluation.

Purification	With	Without
Misclassified	0.1%	9.8%
False negatives	5.2%	2.1%

Table 5: Cluster quality under unsampled evaluation.

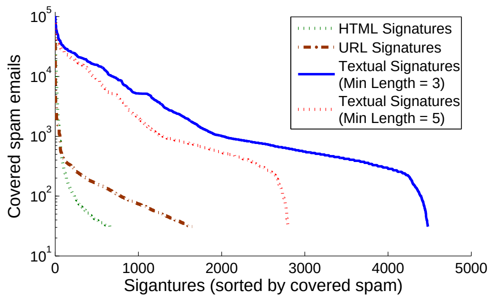

Figure 2: Coverage of spam in DEPT by HTML, URL, and textual signatures.

For the RELAY dataset, only 33.2% of spam contain HTML structures and 12.2% of spam could be detected by HTML signatures. Most of the remaining spam do share the same HTML trees, but the tree structures are too simple to form a signature. Therefore, as shown in Table 8, the TPR of HTML-based detector is only 12%. Furthermore, 69.1% of spam contain URLs (the remaining spam have telephone numbers, SkyeIDs, or physical addresses). However, SCA does not generate any URL signature since there are no observed obfuscations[6]. Such fixed domains without any obfuscation are not good indicators for spam since it is entirely possible that fixed domains occur in legitimate campaigns such as newsletters. For the ENRON dataset (Table 9), no HTML/URL signature was generated since very few ham in the trace contain HTML tags or URLs.

6.3 Textual Signatures

We now evaluate the overall quality of textual signatures in performing spam detection. For the DEPT and the RELAY trace, respectively, SCA outputs 4,480 and 2,095 textual signatures when the minimum signature length is 3, and 2,798 and 833 textual signatures when the minimum signature length is 5. Figure 2 shows the distribution of textual signature coverage (for DEPT), which also exhibits heavy-tail distribution. Table 10 lists a few examples of textual signatures that can be either product name, (fake) addresses, or meaningless chaff injected to deceive the supervised classifiers.

Table 6 and Table 8 also show the accuracy of textual signatures for DEPT and RELAY, respectively. Clearly, for both datasets, the detection rates (TPR) of textual signatures are much higher than those of HTML and URL signatures. In particular, textual signatures are capable of identifying nearly all spam in the RELAY trace. The FPR for DEPT under textual signatures, 0.1-0.4%, are still very low. Furthermore, the last two columns of Table 6 and Table 8 demonstrate the effectiveness of unsupervised *Ensemble* learning: for DEPT, by combing three types of signatures, SCA achieves a TPR of 94.9%, a 14% improvement compared to only using textual signatures.

Table 9 shows the detection results for the ENRON trace from which no signatures were generated due to the high-entropy nature of legitimate emails. For each segment, the bisect k-means algorithm generates a large number of small clusters with low quality. Then all clusters are filtered in the purification procedure.

7. ONLINE UNSUPERVISED DETECTION

In this section, we present the detailed design and evaluation of the *online* version of SCA which exploits campaigns and their signatures obtained from historically received emails. In §7.1 and §7.2, we describe the online learning component and the detection component of the system, focusing on the difference from the offline scheme. We then present the accuracy results in §7.3 and the performance results in §7.4.

7.1 Learning

Figure 3 shows the online learning and detection process of SCA. The learning component uses the text-mining framework (§4.1) to process the input email stream at the unit of one segment for low processing time, and generates signatures based on a sliding window of w most recent segments. When a full segment is ready, it is processed by the per-segment routines described from §4.1.1 to §4.1.5 to obtain signatures for that new segment. Next, the new segment, together with the previous $w-1$ segments, are agglomerated (§4.1.6) to generate the final textual signatures. The signature window of w recent segments ensures that the signatures are up-to-date. Obviously each segment needs to be processed only once even though it stays in the sliding window w times. The overhead of generating HTML and URL signatures is linear *w.r.t.* the number of emails. Therefore, we do not employ the "split-and-merge" scheme for HTML and URL signatures. But the signature win-

[6]Formally, in Algorithm 2 described in §4.2, the URLEntropy, controlled by ε_{url} , is zero or close to zero.

Table 6: Offline results for DEPT. "$\surd_3$" and "$\surd_5$" correspond to textual signatures with min lengths of 3 and 5, respectively.

HTML signatures	√		√			√	√
URL signatures		√	√			√	√
Text signatures				$\surd_3$	$\surd_5$	$\surd_3$	$\surd_5$
Have SA Records	72.5%	72.5%	72.5%	72.5%	72.5%	72.5%	72.5%
SA: spam SCA: Spam	31.2%	21.4%	42.9%	63.9%	56.7%	75.8%	72.3%
SA: Ham, SCA: Ham	19.4%	19.8%	19.0%	19.3%	19.5%	18.1%	18.0%
SA: Spam, SCA: Ham	48.5%	58.2%	36.8%	15.4%	22.7%	4.2%	8.0%
SA: Ham, SCA: Spam	0.9%	0.6%	1.3%	1.4%	1.1%	1.8%	1.7%
Misclassified by SA	99%	99%	99%	96%	98%	96%	97%
Misclassified by SCA	1%	1%	1%	4%	2%	4%	3%
No SA Records	27.5%	27.5%	27.5%	27.5%	27.5%	27.5%	27.5%
False Positive Rate (FPR)	<0.1%	<0.1%	0.1%	0.3%	0.1%	0.4%	0.3%
True Positive Rate (TPR)	39.8%	27.4%	54.6%	80.9%	71.8%	94.9%	90.2%

Table 7: Example URL signatures.

`http://*.*.interia.*` (88175 spam)	`http://*.pochta.*/*.*` (6776 spam)
`http://jfedcbue.w.interia.pl`	`http://rogerswsinclair010.pochta.ru/haxu.htm`
`http://xiureuyo.fm.interia.pl`	`http://susiewait0321.pochta.ru/tofu.html`
`http://nbbhobdi.eu.interia.pl`	`http://harrysatodd7269.pochta.ru/vucfo.html`

dow containing $n \cdot w$ recent emails (n is the segment size) is still maintained by the system to keep signatures up-to-date.

7.2 Detection

The visibility problem faced by SCA as described in §3.3 is further exacerbated in online spam detection, as SCA is expected to detect a new campaign upon its onset, *i.e.,* upon witnessing the first batch of spam from that campaign. In the following, we first present the basic online detection scheme of SCA. We then present an optimization to the basic scheme that effectively reduces the false negative ratio.

Spam detection is always performed on a per-email basis. An incoming email is first checked against HTML and URL signatures, then the more expensive textual signatures if none of HTML/URL signatures match the email. Apparently, for each email, performing textual matching for hundreds of signatures may incur non-trivial computational overhead. Such performance overhead could be alleviated by exploiting the heavy-tail distribution of campaign sizes, *i.e.,* a large fraction of spam belong to a small number of campaigns (Figure 2). SCA periodically (every 1K emails) reorders the textual signatures based on their hit counts, *i.e.,* the number of spam they matched. We observe that with dynamic signature reordering enabled, up to 47% emails could be tested against no more than 10 signatures for the DEPT dataset.

7.2.1 Self-Maintained IP Blacklisting

To alleviate the potential effects of the visibility problem while maintaining SCA as an unsupervised learning scheme, we enhance the online detection process of SCA by adding a self-maintained IP blacklist of end hosts that have originated a high spam-to-ham ratio. This is motivated by the well-known *bimodal* behavior of email sources (The vast majority of IP addresses originate either a spam-to-ham ratio or ham-to-spam ratio close to 0, *i.e.,* their behaviors are either consistently malicious or consistently benign [45]) and by the fact that spamming IPs are known to participate in multiple spam campaigns launched in similar time [30]. For each IP, SCA actively maintains the spam ratio (the ratio of emails that matched some campaign signatures, in the w recent segments). In the detection phase, when an email does not match any of the HTML/URL/textual signatures, SCA checks the spam ratio or the source IP, and flags the email as spam if the ratio is close to 1.

7.3 Detection Accuracy

Figure 4 shows the online detection results without blacklisting for the DEPT trace. We observe that increasing the signature window size improves the TPR, until the window size reaches beyond 15 segments. The FPR of the online detection remains at no more than 0.3% while the TPR, 92.4% (with min signature length of 3), is about 2.5% lower than that achieved by the offline scheme (Table 6). This is expected due to the finite sliding window size that limits the campaigns' visibility to SCA and hence its signature generation. In particular, with a window size w, SCA applies the signature set generated from the past w segments to the emails in the next segment. Such signatures cannot capture any campaign that is starting in that new segment. Similarly Table 11 shows the sliding window in the online version also reduces the TPR from 99.9% to 99.7% for the RELAY trace for minimum signature length of 3.

Figure 5 shows the online detection results for the

HTML	$\surd$			$\surd$
URL		$\surd$		$\surd$
Text			$\surd_3/\surd_5$	$\surd_3/\surd_5$
TP	12.2%	0%	99.9%	99.9%
FN	87.8%	100%	<0.01%	<0.01%
TPR	12.2%	0%	99.9%	99.9%

Table 8: Offline spam detection results for `RELAY` (all emails are spam).

HTML	$\surd$			$\surd$
URL		$\surd$		$\surd$
Text			$\surd_3$	$\surd_3$
TN	100%	100%	100%	100%
FP	0	0	0	0
FPR	0	0	0	0

Table 9: Offline spam detection results for `ENRON` (all emails are ham).

Signature	# Spam
pharmacy industry created	102.7K
autoresponder 2635 meridian pkwy... (29 words)	24.9K
price cialis soft tabs price	53.0K
east minor street emmaus 18098	15.8K
broccoli cauliflower brussels sprouts combination	10.0K
allergies asthma depression	8.7K

Table 10: Example textual signatures.

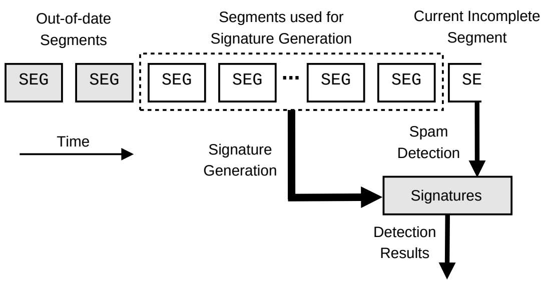

Figure 3: Online learning / detection scheme in SCA

Table 11: Online detection accuracy for `RELAY` with varying signature window sizes, without blacklisting.

Signature Window Size	1	7	15	20
TPR (Min Sig. Len $\geq$ 3)	99.5%	99.7%	99.7%	99.7%
TPR (Min Sig. Len $\geq$ 5)	99.3%	99.4%	99.4%	99.4%

two traces with self-maintained IP blacklisting turned on, as we vary the spam ratio threshold. The spam ratio is defined as the ratio of the number of spam sent (as detected by SCA) over the total number of emails sent in the past, for a given IP address. We observe that blacklisting is highly effective in reducing false negatives. Using a threshold of 0.9, it improves the TPR from 92.4% to 96.5% for the `DEPT` trace and 99.7% to 99.9% for the `RELAY` trace, respectively. The FPR is not affected by blacklisting, and remains at no more than 0.4% for `DEPT` (FPR is always 0 for `RELAY`).

As discussed in §7.2.1, the effectiveness of IP blacklisting on reducing the false negative ratio is explained by the bimodal behavior of the email sources. Figure 6 shows that over 96.9% of the source IPs in the `DEPT` trace originated either only spam or only ham, based on the detection results of SCA. Figure 7 shows that 28% (`DEPT`) and 51% (`RELAY`) of the source IPs in the two traces sent at least two spam.

7.4 Filtering Throughput

To demonstrate that SCA is truly online, we compare its throughput in filtering spam with that of SpamAssassin (SA), the *de-facto* anti-spam solution. Our current prototype of SCA was implemented in about 6K lines of C++ and Perl code. The measurement machine is a commodity Lenovo T60 laptop with 2.0GHz processor and 3GB RAM at 667MHz. Using the parameters in Table 2 and with self-maintained IP blacklisting turned on, we ran SCA by feeding it the `DEPT` dataset as fast as possible. We measured the average processing throughput of SCA to be 18.9 emails/sec (including the time for both campaign generation and spam detection). To measure SA's throughput, we configured SA with all of its four tests: the network test that performs DNSBL lookups, the raw body test that matches mail body against regexes downloaded from the central SA site, the auto-whitelisting and the Bayesian filtering. Then we feed SA with the `DEPT` trace as fast as possible. We optimized the network test using parallel DNSBL lookups that improved SA's email processing throughput from 0.22 emails/sec (sequential DNSBL lookup) to 11.7 emails/sec (200 concurrent DNSBL lookups). These results demonstrate that SCA already outperforms SA in terms of email processing throughput due to the efficient design of its online learning framework.

8. DISCUSSIONS

In this section, we discuss two important issues regarding SCA: *(i)* how to overcome the visibility challenge (*i.e.,* a single organization may not witness enough spam belonging to a campaign); *(ii)* from the perspective of spammers, how difficult it is to evade SCA.

8.1 The Visibility Challenge

Our evaluation in §7 has shown that the online version of SCA achieves 96.5% TPR and no more than 0.4% FPR for the `DEPT` trace. These detection accuracies are already comparable to those of the *de-facto* supervised-learning-based filtering systems such as SpamAssassin (SA) [38, 35, 34].[7] We manually examined the spam belonging to the 3.5% false negatives and

[7]Authors in [38] measured SA's FPR and FNR to vary between 0.02-0.56% and 4.02-5.60%, on 4 department mail

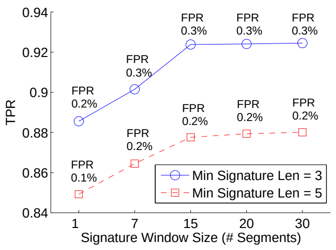

Figure 4: Online detection accuracy for DEPT with varying signature window sizes, w/o blacklisting.

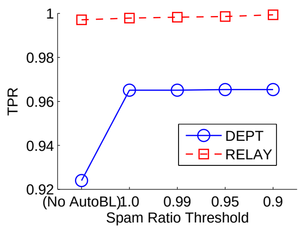

Figure 5: Online detection accuracy with self-maintained IP blacklisting.

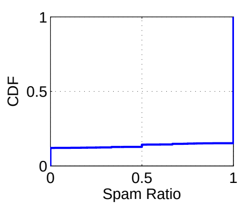

Figure 6: CDF of spam ratio for IPs found in DEPT dataset.

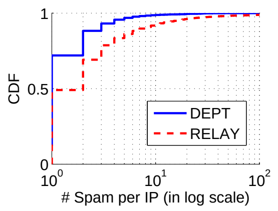

Figure 7: CDF of number of spam per IP for DEPT and RELAY.

found the reason they escaped SCA is indeed due to the visibility challenge discussed in §3.3: SCA did not witness enough of similar spam to identify the campaigns they belong.

First, we note that the DEPT trace is for a single university department covering about 2000 mailboxes. The typical user base for spam filtering at one organization is larger. For example, many universities deploy spam filtering at the university-level mail servers. We expect the visibility of campaigns at this level to be much higher which can potentially further reduce false negatives.

One systematic way to reduce the remaining false negatives is to look up the emails that escaped all signatures and the self-maintained blacklist (§7.2.1) in public DNSBL blacklists. We evaluated this step and found that looking up DNSBLs improves the TPR of SCA on the DEPT trace from 96.5% to 97.1%, and keeps the FPR the same at 0.4% [8]. In comparison, only performing DNSBL lookups for the source IPs in the DEPT trace gives a TPR of 23% and a FPR of 0.1%. This suggests querying DNSBLs alone is far from being sufficient as a spam filtering scheme. While it is arguable if querying DNSBLs is considered unsupervised, it is widely used as one of the filtering steps by many spam filter systems, such as SpamAssassin.

Ultimately, the visibility problem can be overcome by having multiple organizations share their email corpus. Similar approach has been taken by community-based intrusion detection system [32]. To avoid compromising privacy, sharing email corpus can be done anonymously using one-way functions. For example, several organization can agree on using the same one-way function to hash the terms in their email corpus before sharing. The combined corpus can then be used for learning campaigns and extracting campaign signatures. Afterwards, each organization only needs to be concerned with extracting the subset of campaign signatures for which it can map back to unhashed terms. We plan to pursue this in our future work.

8.2 Evasion

It is well-known that spam filtering is a continuous battle between spammers and anti-spam solutions. When a new filtering system gets a step ahead in the arms race, the spammers will exploit new ways to evade it. One possible way to evade SCA is to reduce similarities among spam belonging to the same campaign. This can be achieved by spammers by employing a more complicated template language (*e.g.,* context-sensitive grammar), expanding the vocabulary size by introducing large amount of term-level obfuscation, or confusing the clustering algorithm by increasing the proportion of random chaff. Doing so can potentially reduce the TPR or increase the computation overhead of SCA. However, the spammers also to have pay the price of higher overhead for generating spam and larger spam sizes, which potentially reduces the readability of spam and hence lowers the conversion rates. As suggested by [31], this is probably the reason that template languages have not changed significantly during the past five years.

Another way to evade SCA is to try to generate fewer spams per template, thus not giving SCA enough learning opportunities for signature generation even when SCA is deployed at a large domain (sophisticated spammers knowing the window size w can even reuse the old templates when they are out of the window). However, overall, spammers would need to generate a larger number of templates to maintain the same spam volume. This effect is largely considered a win in the arms race since the efforts or resources of spammers are increased. Further, if a spammer knows the IP-based filtering policy (controlled by u in Table 2), then he/she may launch a campaign originated from no more than u infected hosts. However, each of the fewer u hosts has to send a larger volume of spam, making them easier to be detected. An alternative way to abuse the IP-based filtering policy is to spam using public mail providers. In such case, there are existing techniques to identify

datasets. Also, every version of SA comes with accuracy statistics of the current version against a few public corpora [34]. The FPR and FNR according to statistics in SA vary between 0.06-0.70% and 1.49-7.63%, respectively [35].

[8]We query six public blacklists, and an email is classified as spam if its IP is blocked by at least 2 DNSBLs.

such bot accounts (*e.g.,* BotGraph [52]).

In this study, we focus on generating textual signatures. For HTML and URL signatures, our current prototype performs exact matching for HTML trees and only considers a particular types of URLs consisting both the invariant and the obfuscated components in the URL string. Spammers can exploit this by obfuscating HTML structures and making URLs contain no invariant substring. This can be mitigated by using clustering algorithms for trees [29], and by using sophisticated URL signature generator such as [46, 31]. Finally, SCA in its current form does not work for image-based spam.

9. RELATED WORK

To our knowledge, SCA is the first unsupervised spam filtering system that explicitly exploits online spam campaign identification and achieves accuracy comparable to the *de-facto* supervised spam filters. We have already discussed previous work on supervised spam filtering in §2 and on unsupervised spam filtering in §3. Below, we briefly summarize previous work on spam campaign identification, then compare our system with Judo [31], a recently proposed anti-spam scheme aiming at inferring underlying templates which are then used to generate signatures.

9.1 Existing Campaign Identification Techniques

Existing work on campaign identification fall into four categories. (1) A number of work identified campaigns via manual classification and exact matching [23, 21, 30, 25], based on the common templates used, the common URLs or URL-groups embedded in the email bodies, or the common, finally redirected web pages pointed by the URLs. (2) The second category includes work that extract token streams from URLs or a portion of message body and use MD5 or SHA1 to generate a hash of the token stream as the campaign signature [49, 20]. Such hashing-based signatures are highly sensitive to the slightest obfuscation or noise. (3) The third category includes approaches that use text shingling [4] for spam campaign identification (*e.g.,* [53]). It is known [7] that shingling suffers poor scalability due to its quadratic time complexity, and poor accuracy, especially on small documents (*e.g.,* less than 4KB). (4) The final category includes automated systems like [46, 12, 31] AutoRE [46] is a technique that automatically extracts regular expressions from URLs that satisfy distributed and burstiness criteria as signatures (*e.g.,* `xx.*yy.*zz`). However, this approach tends to yield many uncovered spam since *(i)* a non-trivial fraction of spam do not contain URLs, *(ii)* botnet spam campaigns could last long with non-bursty URLs [30]. We argue that the unsupervised learning should consider the entire message content and complement textual signatures with HTML- and URL-based signatures for added coverage and accuracy. In [5, 6], Calais *et al.* group messages with similar content layout into one campaign, considering partial information, *e.g.,* URL and language.

9.2 Comparing to Judo

Judo [31] is a recent work that aims at inferring the underlying template used to generate the signatures. It examines spam feeds generated by bots and infers the underlying templates by finding out the anchor texts and macros. It also utilizes mail-header knowledge to reverse-engineer the header part of the template. This approach falls in the category of supervised learning since the input (training data) to the system is a clean trace consisting of pure spam generated from templates. Therefore, Judo can achieve very low false positives, and low false negatives, if most received spam are generated by the inferred templates from the same bots. However, Judo imposes the requirement of setting up a bot monitoring infrastructure (like botlab [19]), thus making it difficult to be deployed at any end mail-receiving system (*e.g.,* a university mail server). Furthermore, Judo infers templates from spam produced by bots running in the sandbox. Therefore the coverage of its template-based signatures is limited by the diversity of captured bots and the fraction of templates seen by those particular bot instances. Compared to Judo, SCA is completely unsupervised. It neither requires pre-classified training trace nor any special infrastructures. Instead, by leveraging the low-entropy nature of (campaign-based) spamming, SCA generates high-quality campaign signatures from the raw traces of mixed ham and spam (potentially from several templates and botnets). Also, our system can more quickly adapt to the emergence of new templates or campaigns, again due to its unsupervised nature. In terms of accuracy, our evaluation results indicate that SCA already achieves TPR and FPR comparable to *de-facto* supervised approaches. Finally, we note that Judo and SCA can trivially coexist without any conflict.

10. CONCLUSIONS

We have made two key contributions in this paper. First, we have shown it is feasible to design a completely unsupervised-learning-based spam filtering scheme to achieve spam detection accuracy comparable to those of the *de-facto* supervised-learning-based filtering systems such as SpamAssassin. We elucidated the key observations about today's Internet spam that make such a design feasible and identified the visibility challenge as the main obstacle to the spam detection accuracy of such a filtering system when deployed at individual organizations. Second, we have presented the complete design of SCA, the first unsupervised spam filtering system that explicitly exploits online spam campaign identification, and experimentally shown it achieves accuracy comparable to those of the *de-facto* supervised anti-spam systems. We have further demonstrated its three components, a text-mining based framework for extracting textual campaign signatures, two robust algorithms for extracting HTML and URL signatures, and self-maintained IP blacklist, are all essential and complementary to achieve high detection accuracy and efficiency.

In our ongoing work, we are soliciting and collecting diverse email datasets to further evaluate the detection accuracy of SCA. We plan to study how the size of the organization affects SCA's learning and detection accuracy, and in particular, how much sharing email corpus among multiple organizations is needed to reduce the remaining false negative ratio. We are also integrating SCA with MailAvenger [17] and planning to experiment with it running as a production mail server in our organization.

11. REFERENCES

[1] Enron email dataset. http://www.cs.cmu.edu/∼enron/.

[2] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In *USENIX Security*, 2007.

[3] Microsoft: 3% of e-mail is stuff we want; the rest is spam. http://arstechnica.com/security/news/2009/04/microsoft-97-percent-of-all-e-mail-is-spam.ars.

[4] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Comput. Netw. ISDN Syst.*, 1997.

[5] Pedro Calais, Douglas E. V. Pires, Dorgival Guedes, Wagner Meira Jr., Cristine Hoepers, and Klaus Steding-Jessen. A campaign-based characterization of spamming strategies. In *CEAS*, 2008.

[6] Pedro Calais, Douglas E. V. Pires, Marco Ribeiro, Dorgival Guedes, Wagner Meira Jr., Cristine Hoepers, Marcelo Chaves, and Klaus Steding-Jessen. Spam miner: A platform for detecting and characterizing spam campaigns. In *KDD*, 2009.

[7] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. on Information Systems*, 2002.

[8] James Clark, Irena Koprinska, and Josiah Poon. A neural network based approach to automated e-mail classification. In *IEEE/WIC Intl. Conf. on Web Intelligence*, 2003.

[9] Inside the "ron paul" spam botnet. http://en.wikipedia.org/wiki/Dark_Mailer.

[10] Harris Drucker, Donghui Wu, and Vladimir Vapnik. Support vector machines for spam categorization. In *IEEE Transaction on Neural Networks*, 1999.

[11] K. Gee. Using latent semantic indexing to filter spam, 2003.

[12] Jan Göbel, Thorsten Holz, and Philipp Trinius. Towards proactive spam filtering (extended abstract). In *DIMVA '09: Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2009.

[13] Q3'09 spam & virus trends from postini. http://googleenterprise.blogspot.com/search/label/spam%20and%20security%

[14] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[15] Peter Haider and Tobias Scheffer. Bayesian clustering for email campaign detection. In *ICML*, 2009.

[16] Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, Alexander G. Gray, and Sven Krasser. Detecting spammers with snare: Spatio-temporal network-level automatic reputation engine. In *USENIX Security*, 2009.

[17] Mailavenger. http://www.mailavenger.org.

[18] Ironport: 2008 internet security trends. http://www.ironport.com/securitytrends.

[19] John P. John, Alexander Moshchuk, Steven D. Gribble, and Arvind Krishnamurthy. Studying spamming botnets using botlab. In *NSDI*, 2009.

[20] Aleksander Kolcz and Abdur Chowdhury. Hardening fingerprinting by context. In *CEAS*, 2007.

[21] Maria Konte, Nick Feamster, and Jaeyeon Jung. Dynamics of online scam hosting infrastructure. In *PAM*, 2009.

[22] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. On the spam campaign trail. In *LEET*, 2008.

[23] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamcraft: An inside look at spam campaign orchestration. In *LEET*, 2009.

[24] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes 25:259-284*, 1998.

[25] Fulu Li and Mo-Han Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *CEAS*, 2006.

[26] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes - which naive bayes? In *CEAS*, 2006.

[27] Kazuyuki Narisawa, Hideo Bannai, Kohei Hatano, and Masayuki Takeda. Unsupervised spam detection based on string alienness measures. In *WWW*, 2007.

[28] Kazuyuki Narisawa, Yasuhiro Yamada, Daisuke Ikeda, and Masayuki Takeda. Detecting blog spams using the vocabulary size of all substrings in their copies. In *Workshop of WWW*, 2006.

[29] Andrew Nierman and H. V. Jagadish. Evaluating structural similarity in xml documents. In *Proceedings of the 5th International Workshop on the Web and Databases*, 2002.

[30] Abhinav Pathak, Feng Qian, Y. Charlie Hu, Z. Morley Mao, and Supranamaya Ranjan. Botnet spam campaigns can be long lasting: Evidence, implications, and analysis. In *SIGMETRICS*, 2009.

[31] Andreas Pitsillidis, Kirill Levchenko, Christian Kreibich, Chris Kanich, Geoffrey M. Voelker, Vern Paxson, Nicholas Weaver, and Stefan Savage. Botnet judo: Fighting spam with itself. In *NDSS*, 2010.

[32] Feng Qian, Zhiyun Qian, Z. Morley Mao, , and Atul Prakash. Ensemble: Community-based anomaly detection for popular applications. In *SecureComm*, 2009.

[33] Inside the "ron paul" spam botnet. http://www.secureworks.com/research/threats/ronpaul/.

[34] Spamassassin corpora. http://wiki.apache.org/spamassassin/RescoreMassCheck310.

[35] Spamassassin statistics. http://mirror.olnevhost.net/pub/apache/spamassassin/source/Mail-SpamAssassin-3.2.5.tar.gz.

[36] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *Info. Processing and Management*, 1988.

[37] Fabrizio Sebastiani. Machine learning in automated text categorization. In *ACM Computing Surveys*, 2002.

[38] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In *MALWARE 2008*, Oct. 2008.

[39] Henry Stern. A survey of modern spam tools. In *CEAS*, 2008.

[40] Henry Stern. The rise and fall of reactor mailer. In *The MIT Spam conference*, 2009.

[41] Messagelabs intelligence: Reputable sources are cyber criminals favored resources; spammers work by us clocks. http://www.messagelabs.com/download.get?filename=MLIReport_2009_05_May_FINAL.pdf.

[42] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* Addison-Wesley, 2006.

[43] Konstantin Tretyakov. Machine learning techniques in spam filtering. In *Data Mining Problem-oriented Seminar, MTAT.03.177*, 2004.

[44] Takashi Uemura, Daisuke Ikeda, and Hiroki Arimura. Unsupervised spam detection by document complexity estimation. In *the 11th Intl. Conf. on Discovery Science*, 2008.

[45] Shobha Venkataraman, Subhabrata Sen, Oliver Spatscheck, Patrick Haffner, and Dawn Song. Exploiting network structure for proactive spam mitigation. In *Usenix Security*, 2007.

[46] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: Signatures and characteristics. In *SIGCOMM*, 2008.

[47] Jun-Ming Xu, Giorgio Fumera, Fabio Roli, and Zhi-Hua Zhou. Training spamassassin with active semi-supervised learning. In *CEAS*, 2009.

[48] Yiming Yang and Jan Pedersen. A comparative study on feature selection in text categorization. In *Prof. of ICML*, 1997.

[49] Chun Chao Yeh and Chia Hui Lin. Near-duplicate mail detection based on url information for spam filtering. *LNCS*, 3961, 2006.

[50] Kenichi Yoshida, Fuminori Adachi, Takashi Washio, Hiroshi Motoda, Teruaki Homma, Akihiro Nakashima, Hiromitsu Fujikawa, and Katsuyuki Yamazaki. Density-based spam detector. In *KDD*, 2004.

[51] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. In *ACM Transactions on Asian Language Information Processing*, 2004.

[52] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. Botgraph: Large scale spamming botnet detection. In *NSDI*, 2009.

[53] Li Zhuang, John Dunagan, Daniel R. Simon, Helen J. Wang, Ivan Osipkov, Geoff Hulten, and J.D. Tygar. Characterizing botnets from email spam records. In *LEET*, 2008.