AUTOMATIC IDENTIFICATION AND REMOVAL OF LOW QUALITY ONLINE INFORMATION

A Thesis
Presented to
The Academic Faculty

by

Steve Webb

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2008

AUTOMATIC IDENTIFICATION AND REMOVAL OF LOW QUALITY ONLINE INFORMATION

Approved by:

Calton Pu, Advisor
College of Computing
Georgia Institute of Technology

Ling Liu
College of Computing
Georgia Institute of Technology

Mustaque Ahamad
College of Computing
Georgia Institute of Technology

Shyhtsun Felix Wu
Department of Computer Science
University of California, Davis

Nick Feamster
College of Computing
Georgia Institute of Technology

Date Approved: 13 August 2008

For My Mother

ACKNOWLEDGEMENTS

The quest to obtain a Ph.D. is often characterized as a rollercoaster with numerous peaks and valleys, and as any Ph.D. candidate will tell you, I have experienced my fair share of those peaks and valleys. Fortunately, throughout the highest of highs and the lowest of lows, I have been surrounded by people that have offered constant encouragement and guidance.

First and foremost, I am eternally grateful for the undying support that I have received from my family. I am particularly thankful for my mother, Anne, and my grandmother, Anna Ross. These two women have always been the most important people in my life, and without their unconditional love and support, I would not be the man I am today. Thank you both for everything you have done for me, and know that I will always love you. I am also thankful for my father, John. He taught me the importance of hard work and dedication, and I am truly grateful for those lessons. I love you, Dad. The rest of my family is equally amazing, and I truly appreciate their love and support. Beyond my immediate family, I am also blessed to be an honorary member of the Hammond and Little families. We might not be connected by blood, but I still love you with all of my heart.

At Georgia Tech, my advisor, Calton Pu, was a constant source of inspiration. Regardless of what graduate school or life threw at me, Calton was always there to put things in their proper perspective. In my very first meeting with Calton, he congratulated me for choosing personal and intellectual growth over the shortsighted allure of immediate financial gains. At the time, I have to admit that I thought he was crazy. But now, I understand exactly what he meant, and I cherish the journey we completed together. I am also grateful for Ling Liu's guidance. At every one of my DISL or DoI presentations, I could always rely on her to make interesting observations about my work, and ultimately, her insights greatly enhanced my dissertation research. Leo Mark and H. Venkateswaran were also very positive influences at Tech. I could always expect a smile and a wave when I saw them on campus.

I am indebted to Calton, Ling, Mustaque Ahamad, Nick Feamster, and Shyhtsun Felix Wu for reading and commenting on my dissertation. Their feedback was incredibly valuable, and I truly appreciate it. In addition to my committee members, I was fortunate enough to be exposed to a number of amazing people throughout my graduate career. Galen Swint and Kang Li were essentially my second and third advisors during my first year at Tech, and I appreciate every piece of advice they gave me. Subramanyam Chitti was an amazing collaborator, and I genuinely enjoyed every conversation, all-nighter, and Tin Drum trip we shared together. Jinpeng Wei and Younggyun Koh will always have a special place in my heart because we all endured the Systems Qualifier together. Li Xiong was an amazing officemate and an even better Master Shake. James Caverlee was one of the primary reasons I came to campus every day, and the three years we shared an office together were among the highlights of my life. Tim Garcia, Danesh Irani, and Qinyi Wu were also amazing officemates. I will miss tossing Frisbees with them all over the Klaus Building at all hours of the night. Finally, I am very thankful for the assistance I received from Deborah Mitchell, Mary Claire Thompson, and Susie McClain. I cherish my friendships with all of you, and I hope we will always stay in touch.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The creation of the Internet has fundamentally changed the way we communicate, conduct business, and interact with the world around us. Specifically, the Internet has given us access to a host of information-rich environments such as email systems, the World Wide Web, and social networking communities, which provide information consumers with an unprecedented amount of freely available information. However, the openness of these environments has also made them vulnerable to a new class of attacks called Denial of Information (DoI) attacks. Attackers launch these attacks by deliberately inserting low quality information into information-rich environments to promote that information or to deny access to high quality information. These attacks directly threaten the usefulness and dependability of online information-rich environments, and as a result, an important research question is how to automatically identify and remove this low quality information from these environments.

In this thesis research, we focus on answering this important question by countering DoI attacks in three of the most important information-rich environments: email systems, the World Wide Web, and social networking communities. For each environment, we perform large-scale data collection and analysis operations to create massive corpora of low and high quality information. Then, we use our collections to identify characteristics that uniquely distinguish examples of low and high quality information. Finally, we use our characterizations to create techniques that automatically detect and remove low quality information from online information-rich environments.

The first contribution of this thesis research is a set of techniques for automatically recognizing and countering various forms of DoI attacks in email systems. Initially, we show experimentally that statistical classifiers are able to distinguish between large corpora of low quality email messages (i.e., spam email) and high quality email messages (i.e., legitimate email) with a high degree of accuracy. Then, we design a new DoI attack that uses

camouflaged messages (i.e., spam messages that mask their spam content with high quality content that is stolen from legitimate messages) to significantly degrade the performance of these statistical classifiers. Next, we show that the accuracy of most of the classifiers can be restored by retraining the classifiers with camouflaged messages in their training sets.

Unfortunately, we quickly discover that the classifier retraining process is only temporarily effective against camouflaged messages. Due to the constantly evolving nature of spammers, it is only a matter of time before retrained classifiers are vulnerable to the next generation of attacks (i.e., new camouflaged messages), and as a result, spam producers and information consumers quickly become entrenched in a spam arms race. To break free of this arms race, we propose two solutions. One solution involves refining the statistical learning process by associating disproportionate weights to spam and legitimate features, and the other solution leverages the existence of non-textual email features (e.g., URLs) to make the classification process more resilient against attacks.

The second contribution of this thesis is a framework for collecting, analyzing, and classifying examples of DoI attacks in the World Wide Web. Our first observation in this domain is that research progress has been limited by the lack of a publicly available Web spam corpus. To alleviate this situation, we propose a fully automatic Web spam collection technique that leverages the URLs found in email spam messages to extract large Web spam samples. Using this new approach, we created the Webb Spam Corpus – a first-of-its-kind, large-scale, and publicly available Web spam data set. The corpus consists of nearly 350,000 Web spam pages, making it more than two orders of magnitude larger than any other previously cited Web spam data set.

Using the Webb Spam Corpus, we perform the first large-scale characterization of Web spam using content and HTTP session analysis. Our content analysis shows that the rate of duplication among Web spam pages is twice the duplication rate for legitimate Web pages, and it identifies five important categories of Web spam: Ad Farms, Parked Domains, Advertisements, Pornography, and Redirection. Additionally, our HTTP session analysis illustrates two important trends. First, the IP addresses that actually host the Web spam pages are concentrated in two narrow ranges (63.* – 69.* and 204.* – 216.*). Second,

significant overlaps exist among the session header values.

The results of our HTTP session analysis are particularly interesting because they imply that Web spam can be identified without the heavyweight approaches used in previous research. Rather than analyze the contents of Web pages and the link structure of the Web graph, we present a lightweight, predictive approach to Web spam classification that relies exclusively on HTTP session information (i.e., hosting IP addresses and HTTP session headers). Specifically, we use HTTP session information to train classification algorithms that distinguish between spam and legitimate Web pages. Then, by incorporating these classifiers into HTTP retrieval operations, we are able to detect Web spam pages before the actual content transfer.

The final contribution of this thesis research is a collection of techniques that detect and help prevent DoI attacks within social environments (particularly social networking communities). Over the past few years, these communities have experienced unprecedented growth, and as a result, individuals are attaching an increasing amount of value to their online personas. Unfortunately, the rising importance and prominence of these communities have also made them prime targets for two distinct DoI attack classes: traditional attacks, which have plagued users in other information-rich environments (e.g., email spam, comment spam, etc.), and social-specific attacks, which leverage deceptive profiles (e.g., rogue advertising profiles and impersonating profiles) to accomplish their objective.

First, we provide detailed descriptions for each of these attack classes, and we show that the continued success of social networking communities is contingent upon their ability to mitigate the risks associated with these attacks. Then, we focus our attention on social spam. Due to a lack of data, very little is known about social spammers, their level of sophistication, or their strategies and tactics. Hence, we developed a novel technique for capturing examples of social spam, and we used our collected data to perform the first characterization of social spammers and their behaviors. Concretely, we introduce social honeypots for tracking and monitoring social spam, and we report the results of an analysis performed on spam data that was harvested by our social honeypots. Based on our analysis, we find that the behaviors of social spammers exhibit recognizable temporal and

geographic patterns and that social spam content contains various distinguishing characteristics. These results are quite promising and suggest that our analysis techniques may be used to automatically identify social spam.

CHAPTER I

INTRODUCTION

The advent of the Internet has generated a proliferation of online information-rich environments, which provide an enormous amount of value for information consumers. Email systems now facilitate the majority of online communication; the World Wide Web serves as the primary portal for sharing information, and social networking communities are allowing individuals to exchange information in a variety of new and exciting ways. Each of these environments exhibits a degree of openness that allows individuals to obtain information freely and efficiently. However, the open nature of these environments also makes them vulnerable to a new wave of attacks known as Denial of Information (DoI) attacks [2, 36], which are characterized by the deliberate insertion of low quality information (or noise) into information-rich environments.

DoI attacks are the information analog to Denial of Service (DoS) attacks. Just as DoS attacks flood services with syntactically correct requests to degrade the Quality of Service (QoS) of those services, DoI attacks flood information-rich environments with syntactically correct noise data to degrade the Quality of Information (QoI) in those environments. The QoI of an environment serves to measure the relevance and usefulness of the information in that environment, as it relates to the environment's information consumers. High quality information is highly relevant to the needs of information consumers, providing significant value to those consumers. Conversely, low quality information is irrelevant to the needs of information consumers, and it contributes little to no value to consumers.

Attackers primarily perform DoI attacks to accomplish one of two goals: promotion of particular ideals by means of deception and denial of access to high quality information. Neither of these goals is beneficial for information consumers, and as a result, consumers seek to eliminate DoI attacks from information-rich environments. However, as information consumers deploy countermeasures for DoI attacks, attackers quickly evolve their techniques

to ensure the continued success of the attacks. Thus, an adversarial tension exists between attackers and information consumers, which forces proposed solutions to DoI attacks to be resilient against a constantly evolving adversary.

In this thesis research, we identify various forms of DoI attacks in three very distinct but inter-related information-rich environments: email systems, the World Wide Web, and social networking communities. To counter these attacks, we propose various techniques for automatically identifying and removing low quality information in each of these rather unique domains. These techniques utilize state of the art classification algorithms and informative features that are difficult for adversaries to manipulate.

The first contribution of this thesis research is a set of techniques for automatically recognizing and countering various forms of DoI attacks in email systems. Email systems were among the first information-rich environments to gain wide spread acceptance by information consumers, and as a result, they were also one of the first systems to be targeted by DoI attacks. To counter these attacks, many information consumers began relying on statistical spam classifiers to automatically identify and remove low quality information (i.e., spam email) from their inboxes. Consequently, many previous researchers have investigated the effectiveness of these classifiers on small corpora of email messages.

To extend previous research, we performed a large-scale experimental evaluation of statistical spam classifier effectiveness, which provides two valuable insights. First, the evaluation illustrates the importance of using large corpora when evaluating classifiers to ensure consistently reliable results. Second, the evaluation shows that classifiers are able to distinguish between large corpora of low quality email messages and high quality email messages with a high degree of accuracy. Although these results are encouraging, we observed a potential problem with the assumptions underlying these classifiers, which we believed could be exploited by spammers. Specifically, we hypothesized that spammers could construct camouflaged messages (i.e., spam messages containing spam content that is camouflaged by legitimate content) to bypass statistical spam classifiers. After performing a number of attacks against our classifiers using camouflaged messages, we concluded that our hypothesis was correct. A spammer can significantly degrade the performance of classifiers with

camouflaged spam content. Fortunately, we are able to restore most of the accuracy for the classifiers by retraining them to identify camouflaged messages as spam.

While performing our classifier evaluations, we identified a clear tension between spam producers and information consumers. Spam producers are constantly evolving their techniques to ensure their spam messages are delivered, and information consumers are constantly evolving their countermeasures to ensure they don't receive spam messages. This ongoing struggle is often modeled as an arms race, and to help characterize it experimentally, we began investigating the evolution of the construction techniques used by spammers. As a result of this investigation, we identified numerous examples of spam construction techniques that were ineffective due to various countermeasures, and we also identified many examples of techniques that were thriving, despite the presence of seemingly effective countermeasures (i.e., they were coexisting with the countermeasures).

Based on the results of our evolutionary study, we began to question the validity of retraining as a solution for camouflaged messages. Since spammers continually evolve their techniques, we believed they would also evolve their camouflaged messages, making them more sophisticated over time. Thus, we evaluated the effectiveness of retraining against more advanced camouflaged messages, and we quickly discovered that the classifier retraining process is only temporarily effective against camouflaged messages. As information consumers evolve and retrain their classifiers, spammers construct new camouflaged messages, which represent a new generation of attacks. This process continues until both parties are firmly entrenched in a spam arms race. Fortunately, in this thesis, we propose two solutions that allow information consumers to break free of this arms race. The first solution alters the statistical classifier training process by associating disproportionate weights to spam and legitimate features, and the second solution incorporates various non-textual email features (e.g., URLs) to significantly enhance the robustness of the spam classification process.

The second contribution of this thesis is a framework for collecting, analyzing, and classifying examples of DoI attacks in the World Wide Web. Just as email spam has negatively impacted the user messaging experience, the rise of *Web spam* is threatening to severely degrade the quality of information on the World Wide Web. Fundamentally,

Web spam is designed to pollute search engines and corrupt the user experience by driving traffic to particular spammed Web pages, regardless of the merits of those pages. Hence, we present various techniques for automatically identifying and removing these pages from the Web.

First, we identify an interesting link between email spam and Web spam, and we use this link to propose a novel technique for extracting large Web spam samples from the Web. Then, we present the Webb Spam Corpus – a first-of-its-kind, large-scale, and publicly available Web spam data set that was created using our automated Web spam collection method. The corpus consists of nearly 350,000 Web spam pages, making it more than two orders of magnitude larger than any other previously cited Web spam data set. To help motivate the usefulness of this corpus, we also identify several application areas where the Webb Spam Corpus may be especially helpful. Interestingly, since the Webb Spam Corpus bridges the worlds of email spam and Web spam, we note that it can be used to aid traditional email spam classification algorithms through an analysis of the characteristics of the Web pages referenced by email messages.

After presenting the Webb Spam Corpus, we leverage its pages to perform the first large-scale characterization of Web spam using content and HTTP session analysis techniques. Our content analysis results are consistent with the hypothesis that Web spam pages are different from legitimate Web pages, showing far more duplication of physical content and URL redirections. Additionally, our content analysis offers a categorization of Web spam pages, which includes Ad Farms, Parked Domains, Advertisements, Pornography, and Redirection. Next, an analysis of session information collected during the crawling of the Webb Spam Corpus shows significant concentration of hosting IP addresses in two narrow ranges as well as significant overlaps among session header values. These findings suggest that new content and HTTP session analysis techniques may contribute a great deal towards future efforts to automatically distinguish spam Web pages from legitimate Web pages.

To defend against Web spam, most previous research analyzes the contents of Web pages and the link structure of the Web graph. Unfortunately, these heavyweight approaches

require full downloads of both legitimate and spam pages to be effective, making real-time deployment of these techniques infeasible for Web browsers, high-performance Web crawlers, and real-time Web applications. Leveraging the results of our HTTP session analysis, we present a lightweight, *predictive* approach to Web spam classification that relies exclusively on HTTP session information (i.e., hosting IP addresses and HTTP session headers). Concretely, we built an HTTP session classifier based on our predictive technique, and by incorporating this classifier into HTTP retrieval operations, we are able to detect Web spam pages before the actual content transfer. As a result, our approach protects Web users from Web-propagated malware, and it generates significant bandwidth and storage savings. By applying our predictive technique to a corpus of almost 350,000 Web spam instances and almost 400,000 legitimate instances, we were able to successfully detect 88.2% of the Web spam pages with a false positive rate of only 0.4%. These classification results are superior to previous evaluation results obtained with traditional link-based and content-based techniques. Additionally, our experiments show that our approach saves an average of 15.4 KB of bandwidth and storage resources for every successfully identified Web spam page, while only adding an average of 101μs to each HTTP retrieval operation. Therefore, our predictive technique can be successfully deployed in applications that demand real-time spam detection.

The final contribution of this thesis research is a collection of techniques that detect and help prevent DoI attacks within social environments (particularly social networking communities). Online social networking communities are connecting hundreds of millions of individuals across the globe and facilitating new modes of interaction. Due to their immense popularity, an important question is whether the high quality information in these communities is accessible by their users. In this thesis research, we address this question and show that social networking communities are susceptible to numerous attacks. Specifically, we identify two attack classes: traditional attacks that have been adapted to these communities (e.g., malware propagation, spam, and phishing) and new attacks that have emerged through malicious social networking profiles (e.g., rogue advertising profiles and impersonating profiles). Concretely, we describe examples of these attack types that are

observable in MySpace, which is currently the most popular social networking community.

After we describe the various security threats that exist in social environments, we focus our attention on social spam. Unfortunately, little is known about social spammers, their level of sophistication, or their strategies and tactics. Thus, in this thesis research, we offer a novel technique for capturing examples of social spam, and we provide the first characterization of social spammers and their behaviors. Concretely, we make two contributions: (1) we introduce *social honeypots* for tracking and monitoring social spam, and (2) we report the results of an analysis performed on spam data that was harvested by our social honeypots. Based on our analysis, we find that the behaviors of social spammers exhibit recognizable temporal and geographic patterns and that social spam content contains various distinguishing characteristics. These results are quite promising and suggest that our analysis techniques may be used to automatically identify social spam.

The remainder of this thesis is organized as follows:

- **Chapter 2: Preliminaries** – We review fundamental concepts that are repeatedly used in future chapters about email spam. These topics include an overview of statistical classification, brief descriptions of popular classification algorithms, and a description of feature selection.

- **Chapter 3: The Importance of Large Corpora for Email Spam Classification** – In this chapter, we present a large-scale evaluation of a Naïve Bayesian classifier's effectiveness against email spam. This evaluation showcases the importance of using large corpora when performing classifier evaluations, and it offers guidelines for future spam classifier experiments.

- **Chapter 4: Large-Scale Evaluation of Email Spam Classifiers** – Following the guidelines set forth in the previous chapter, we perform a large-scale evaluation of four popular email spam classifiers. First, we show that these classifiers are able to successfully distinguish between spam and legitimate messages. Then, we introduce an effective attack against these classifiers that uses camouflaged messages to confuse the classification process. Finally, we offer an effective solution that retrains the classifier

to identify the camouflaged messages as spam.

- **Chapter 5: Evolutionary Characteristics of Email Spam** – In this chapter, we present the results of an evolutionary study on the construction techniques utilized in email spam messages. This study focuses on two trends: extinction, where the population of messages employing a given technique falls to zero or near zero, and co-existence, where the population maintains a consistent level or grows.

- **Chapter 6: Defending Classifiers Against Camouflaged Email Spam** – Based on the evolutionary properties of email spam, we revisit the camouflaged spam message problem to determine if retraining is a viable long-term solution. We quickly realize that retraining is only temporarily effective because new, randomized camouflaged content is easily generated by spammers. To counter this problem, we offer a solution that changes the current methodology for training email spam classifiers.

- **Chapter 7: Integrating Diverse Email Spam Filtering Techniques** – Continuing the theme of the previous chapter, we offer an integrated approach to spam filtering that is more robust than individual techniques against email spam and camouflaged content. Concretely, we describe a new type of URL-based filtering, and we highlight the benefits of integrating diverse techniques to create a multi-layered defense against spam.

- **Chapter 8: Using Email Spam to Identify Web Spam Automatically** – In this chapter, we present a fully automated technique for collecting Web spam examples that leverages the link between email spam and Web spam. Then, we describe the Webb Spam Corpus – a large-scale and publicly available Web spam data set that was created with our automated technique.

- **Chapter 9: Characterizing Web Spam Using Content and HTTP Session Analysis** – Using the Webb Spam Corpus, we provide the first large-scale characterization of Web spam using content and HTTP session analysis techniques. Based on this characterization, we observe various characteristics that uniquely distinguish

Web spam pages from legitimate pages.

- **Chapter 10: Predicting Web Spam With HTTP Session Information** – Based on the results of our HTTP session analysis, we present a predictive approach to Web spam classification that relies exclusively on HTTP session information. Then, we experimentally evaluate this technique and show that it offers superior accuracy and reduced resource requirements when compared to existing approaches.

- **Chapter 11: Exploring the Dark Side of Social Environments** – In this chapter, we describe various DoI attacks that affect social networking communities. Specifically, we identify two attack classes: traditional attacks that have been adapted to these communities and new attacks that have emerged through malicious social networking profiles.

- **Chapter 12: Using Social Honeypots To Identify Spammers** – To help understand different types of social spam and deception, we propose a novel technique for harvesting deceptive spam profiles that utilizes social honeypots. Then, we provide a characterization of spam profiles using the data we collected with our social honeypots. This characterization illustrates various distinguishing characteristics of spam profiles, and it suggests that our analysis techniques can be used to automatically detect social spam.

- **Chapter 13: Conclusions and Future Research Directions** – We conclude by providing a summary of our thesis research contributions and offering various directions for future research.

CHAPTER II

PRELIMINARIES

In this chapter, we review a few of the fundamental concepts that appear frequently in our discussions about email spam. These topics include an overview of statistical classification, descriptions of three commonly used statistical classification algorithms (Naïve Bayes, Support Vector Machines, and LogitBoost), and a brief explanation of dimensionality reduction.

2.1 Statistical Classification of Email Spam

Email classification can be characterized as the problem of assigning a boolean value ("spam" or "legitimate") to each email message m in a collection of email messages M. More formally, the task of spam classification is to approximate the unknown target function $\Phi : M \rightarrow \{spam, legitimate\}$, which describes how messages are to be classified, by means of a function $\phi : M \rightarrow \{spam, legitimate\}$ called the classifier (or model), such that Φ and ϕ coincide as much as possible.

In the machine learning approach to email classification, a general inductive process (also called the learner) automatically builds a classifier by observing a set of documents that are manually classified as "spam" or "legitimate" (these documents are often called a training set). In machine learning terminology, this classification problem is an example of supervised learning because the learning process is supervised by the knowledge of the category of each message that is used during training.

Different learning methods have been explored by the research community for building spam classifiers (also called spam filters). In our email spam experiments, we focus on three learning algorithms: Naïve Bayes [110], Support Vector Machines (SVM) [147], and LogitBoost [61]. In the following sections, we will briefly summarize the important details of each of these algorithms.

2.1.1 Naïve Bayes

The Naïve Bayes learning algorithm is one of the simplest and most widely used statistical learning solutions. Additionally, it has been utilized in a number of previous spam filtering evaluations [6, 7, 8, 9, 84, 108, 117, 127, 156]. Given a message m, represented as a vector of features, we can use Bayes Theorem and the Theorem of Total Probability to calculate the probability that m is either "spam (s)" or "legitimate (l)":

$$P(m = c|\overrightarrow{X} = \overrightarrow{x}) = \frac{P(m=c)\cdot P(\overrightarrow{X}=\overrightarrow{x}|m=c)}{\sum_{c'=\{s,l\}} P(m=c')\cdot P(\overrightarrow{X}=\overrightarrow{x}|m=c)}.$$

The classification of x is the category c that maximizes the above equation. Note that the denominator is the same for all categories and can be ignored. The most important attributes are the a priori probabilities of each category and the a posteriori probabilities of the vectors, given the category, which need to be estimated from the training data. The number of possible vectors (i.e., all combinations of different feature values) is very large (exponential in the number of features), and many of these vectors will be missing or sparsely found in the training data. Thus, estimating the a posteriori probabilities can be quite difficult. To overcome these problems, a simplifying assumption is made, which leads to the Naïve Bayes classifier: all of the features are considered conditionally independent, given the category. With this simplifying assumption, we can rewrite the above equation:

$$P(m = c|\overrightarrow{X} = \overrightarrow{x}) = \frac{P(m=c)\cdot\prod_{i=1}^{n} P(X_i=x_i|m=c)}{\sum_{c'=\{s,l\}} P(m=c')\cdot\prod_{i=1}^{n} P(X_i=x_i|m=c)}.$$

The new equation requires the estimation of a much smaller number of conditional probabilities (linear in the number of features), making their estimation feasible using the training data. Although the independence assumption is overly simplistic, studies in several domains (including spam filtering) have shown Naïve Bayes to be an effective classifier. The probabilities required in this new equation can be calculated as follows:

$$P(m = c) = \frac{N_c}{N_{Training}}$$

and

$$P(X_i = x_i|m = c) = \frac{N_{c\wedge X_i=x_i}}{N_{Training}}.$$

In the above equations, N_c is the number of messages of type c, $N_{c \wedge X_i = x_i}$ is the number of messages of type c with x_i as its i^{th} feature's value, and $N_{Training}$ is the total number of training messages. These probabilities, along with the above equation, constitute the Naïve Bayes classifier.

2.1.2 Support Vector Machines

Support Vector Machines (SVM) is a powerful machine learning technique based on the structured risk minimization principle from Computational Learning Theory. Given a set of messages, represented as feature vectors x_i, the simplest version of the Support Vector Machines learner tries to find a hyperplane in the feature space of these vectors that best separates the two different kinds of messages. More formally, training a Support Vector Machine is equivalent to solving the following optimization problem:

$$\min_{w,b,\xi_i} (\tfrac{1}{2} W^T \cdot W + C \sum_{i=1}^{N} \xi_i)$$
$$\text{subjected to}$$
$$y_i (W^T \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0.$$

In the above equation, W is a weight vector that should be minimized in order to find an optimal linear separating hyperplane in the feature space, and ξ_i are slack variables, which are used together with $C \geq 0$ to find a solution to the above equation in the non-separable cases.

The equation above is representative of the simplest class of SVMs. In general, the feature vectors can be mapped into a higher dimensional space using a non-linear kernel function, and the best separating hyperplane can be found in this higher dimensional space. However, research in text classification [92] has shown that simple linear SVMs usually perform as well as non-linear ones.

2.1.3 LogitBoost with Regression Stumps

LogitBoost belongs to the class of Boosting Classifiers, which are classification algorithms that try to construct a good classifier by repeated training of a weak classifier. Boosting

uses the weak learning procedure to construct a sequence of classifiers $f_1, f_2, \ldots, f_k$, and then, it uses a weighted vote among these classifiers to make a prediction.

The number of classifiers learned by Boosting is equal to the number of iterations for which it is run. Boosting associates a weight with each training instance in each iteration. Initially, all the instances are assigned equal weights, and during each iteration, the weights of the instances are updated so that the weak learner learns a different classification function in each iteration. Intuitively, those training instances, which are misclassified by all of the previous iterations, are assigned the highest weights in the i^{th} round, forcing the i^{th} classifier to concentrate on those instances in the i^{th} round.

2.2 Dimensionality Reduction

One of the major challenges in text classification (of which email classification is an example) is dealing with the enormously large number of features that occur in reasonably sized corpora. Concretely, the typical size of the feature space is greater than 300,000 (denoted 300K for brevity) for a training set of 10K messages. Many of these features are not relevant to the distinction between spam and legitimate messages, and as a result, they introduce noise into the classification process. Thus, using these features while building a classifier would overfit the classifier to the data and introduce errors into the classification process. To avoid this overfitting, a small subset of features is selected from the original feature space, and then, the classifier is trained on this subset. The process of selecting a small number of informative features from a large feature space is called dimensionality reduction (or feature selection).

2.2.1 Information Gain

In our email spam experiments, we used an information theoretic measure called Information Gain [155] to select a user-specified number n of features. Information Gain is defined as follows:

$$IG(f_i, c_j) = \sum_{c \in \{c_j, \overline{c_j}\}} \sum_{f \in \{f_i, \overline{f_i}\}} p(f, c) \cdot log \frac{p(f,c)}{p(f) \cdot p(c)},$$

where f_i is a feature in the feature vector and c_j is one of the classes (i.e., spam or legitimate). Intuitively, the Information Gain of a feature indicates how strongly that feature is associated with a given class. In our experiments, we calculate the Information Gain for each token in the feature space as it relates to spam messages and legitimate messages, respectively. Then, a user-specified number n of tokens with the highest Information Gain scores are selected and used in the training of the filters.

CHAPTER III

THE IMPORTANCE OF LARGE CORPORA FOR EMAIL SPAM CLASSIFICATION

A cornerstone of machine learning research is the consensus that sufficiently large evaluation samples are needed for reproducible results [125]. Natural language processing researchers [13, 14] have also concluded that large corpora provide significantly more reliable experimental results. Consequently, the collection and evaluation of large corpora have been an important part of the machine learning and natural language processing fields. Examples of widely used large corpora include the Penn Treebank corpus [105] and the TREC corpora [114] for speech recognition as well as the Reuters Corpus Volume 1 (RCV1) benchmark [103] for text categorization.

Although machine learning techniques such as the Naïve Bayes classifier form the core of statistical spam filters, many early experimental evaluations of learning spam filters [6, 7, 8, 9, 68, 84, 117, 127, 156] used relatively small email corpora (typically on the order of a few thousand messages selected by a few users) for evaluation. The lack of large evaluation corpora in the early spam filtering literature (i.e., before 2006) is similar to the early experimental results in machine learning, which preceded the availability of widely used large corpora. Analogous to the lessons learned in natural language processing and machine learning, the spam filter evaluations based on small evaluation corpora exhibit a significant lack of consistency. When filters trained with small corpora are used to evaluate test sets taken from other corpora (large or small), the filters show unreliable performance and unpredictable swings between high and low false positive and false negative rates for different corpora.

An application-specific argument used to justify using small corpora in the evaluation of spam filters (e.g., those enumerated in Section 3.2) has been the differences among individuals in judging what is spam and what is legitimate email. We acknowledge the validity of this

argument for client-specific spam filters, which are analogous to using speaker-dependent speech recognition filters and application-specific corpora in machine learning [113]. On the other hand, the analogy also suggests that the small corpora experiments have validity and applicability that is restricted to the small number of users who selected the corpora. The question that arises is whether we are able to train filters that work well for a large number of users.

In analogy to *speaker-independent* speech recognition research, we are interested in the use of large corpora in methodical evaluations of learning spam filters for many users. We believe the training of filters using large corpora will improve the reproducibility of spam filter experiments over diverse test sets of email messages (both large and small). Additionally, filters trained with large corpora have a large potential practical impact on server-based spam filters, which can filter out similar spam messages before many copies reach the clients. One example of server-based filtering is the backbone filtering carried out by members of the Message Anti-Abuse Working Group (MAAWG) [107]. In the first quarter of 2006, they reportedly filtered 80% of the emails they received (a total of 370 billion messages) and allowed 90 billion emails through the Internet backbone to reach 390 million mailboxes.

The main contribution of this chapter is a comparative study of spam filtering experiments with representative large and small email corpora. Our corpora are primarily in English; however, we do not use any language-specific features in our study. Consequently, our techniques should be applicable to the classification of spam messages in other languages as well. Our study shows that small corpora lead to significantly less predictable results when compared to the results obtained by training and testing using large corpora. Concretely, the study compares the experimental evaluation results of a Naïve Bayesian learning spam filter when using small corpora (obtained from sources cited in the current spam filtering literature) and large corpora (collected from publicly available sources such as the SpamArchive spam corpus and the Enron legitimate corpus). When the filter was trained with small corpora, it produced false positive rates varying from 0% to 46% and

false negative rates varying from 3% to 96%; however, when it was trained with large corpora, it only produced false positive rates consistently around 0% and false negative rates varying from 10% to 20%.

The main conclusion derived from our comparative study is that future experimental evaluations of spam filters should use large corpora (on the order of hundreds of thousands of messages from published archives) for the sake of reliability and reproducibility. Although this conclusion may not be surprising to experienced machine learning and natural language processing researchers, it should cause significant methodological changes in experimental spam filtering research since the spam filtering literature is largely dominated by experiments using small corpora.

The rest of the chapter is organized as follows. Section 3.1 describes the representative large corpora and small corpora used in our experiments. Section 3.2 presents a comparative study of spam filtering experiments using our large and small corpora. Section 3.3 continues the comparative study and quantifies, through experimentation, an appropriate size for large corpora in spam filtering research. Section 3.4 outlines related work, and Section 3.5 summarizes our findings.

3.1 Corpora Descriptions

In this section, we describe the representative large corpora and small corpora that were used in our experiments. For each corpus, we detail how it was obtained and the format of its messages. Also, Table 1 summarizes all of the corpora, giving each corpus' name, abbreviation, and message count. Throughout this chpater, we say that a corpus is *biased* if learning filters trained with that corpus produce unreliable results, which are characterized by a high variance in the rate of false positives or false negatives. Various degrees of bias exist for various corpora; however, our contention is that small corpora are more prone to bias than large corpora.

Table 1: Corpus summaries.

Corpus Name	Name Abbreviation	Number of Messages
Small Spam Corpora		
Ling-spam Spam Corpus	L-s Spam	481
SpamAssassin Spam Corpus	SA Spam	2,398
Small Legitimate Corpora		
Ling-spam Legitimate Corpus	L-s Legit.	2,412
Graduate Student Legitimate Corpora		
grad1	grad1	513
grad2	grad2	3,407
grad3	grad3	5,287
Enron Personal Legitimate Corpora		
Sally Beck	beck-s	1,971
Darren Farmer	farmer-d	3,672
Vincent Kaminski	kaminski-v	4,477
Louise Kitchen	kitchen-l	4,015
Michelle Lokay	lokay-m	2,489
Richard Sanders	sanders-r	1,188
William Williams III	williams-w3	2,769
SpamAssassin Legitimate Corpus	SA Legit.	6,451
Large Spam Corpus		
SpamArchive Spam Corpus	SpamArchive	600K
Large Legitimate Corpus		
Enron Legitimate Corpus	Enron	475K

3.1.1 Small Spam Corpora

3.1.1.1 Ling-spam Spam Corpus

The Ling-spam corpora [6, 87], maintained by the Internet Content Filtering Group (i-config) [86], contain a spam corpus of 481 spam messages that were obtained from a personal email account. The format of these messages is slightly unique because their header fields (with the exception of "Subject") were removed by i-config prior to publication. Additionally, four versions of this corpus are published: "bare", "lemm", "lemm_stop", and "stop". We used the "bare" version of this corpus in our evaluation because previous research [6] has shown that the other versions of the corpus, which use a lemmatizer and a stop-list, do not result in significantly different filter performance.

This corpus is potentially a biased sample for two reasons. First, the corpus is extremely small. In fact, it is the smallest corpus we used in our evaluations. Second, the messages found in the corpus were only collected from a single individual's mailbox. Thus, the corpus represents an extremely narrow view of spam messages. Previous research [156] has also expressed these concerns.

3.1.1.2 SpamAssassin Spam Corpus

The SpamAssassin corpora [141], maintained by the SpamAssassin group [140], contain a spam corpus of 2,398 spam messages that were gathered from various SpamAssassin user submissions. All of the messages in this corpus are in a traditional email format, including their original headers.

Although this corpus consists of messages from a number of unique users, it is still potentially biased due to its small size. A couple thousand messages is not nearly enough to be considered a representative sample of spam messages.

3.1.2 Small Legitimate Corpora

3.1.2.1 Ling-spam Legitimate Corpus

In addition to the spam corpus mentioned above in Section 3.1.1.1, the Ling-spam corpora also contain a legitimate corpus of 2,412 legitimate messages that were obtained from a

moderated linguistic mailing list called the Linguist List [146]. The format of these legitimate messages is the same as the format of the Ling-spam spam messages (i.e., the header fields [except "Subject"] are missing). Also, similar to the Ling-spam spam corpus, four versions of this corpus are published, and we chose to use the "bare" version of this corpus for the same reason we chose to use the "bare" Ling-spam spam corpus (as explained in Section 3.1.1.1).

This corpus is potentially biased for two reasons. First, the corpus does not contain enough messages to be considered a representative sample. Second, since this corpus was collected from a linguistic mailing list, the content of the messages is exclusively about issues pertaining to linguistics. Thus, the scope of these messages is extremely narrow and not representative of most legitimate email messages. These concerns are also reiterated in previous research [156].

In addition to the Ling-spam corpora, i-config also maintains the PU123A corpora. We omitted these corpora from our evaluation because they are encoded such that each of the messages' features has been replaced with a unique number. This feature-to-number mapping was done to protect the privacy of the message senders and recipients; however, since those mappings are unpublished for the PU123A corpora, we are unable to convert those messages to the format of our other corpora.

3.1.2.2 Enron Personal Legitimate Corpora

The Enron personal legitimate corpora consist of seven legitimate email collections that were extracted from the Enron corpus [34, 95] by Ron Bekkerman. For a detailed explanation of how these collections were extracted, please consult [16]. Each of the seven collections corresponds to a specific Enron employee's messages: Sally Beck (1,971 messages), Darren Farmer (3,672 messages), Vincent Kaminski (4,477 messages), Louise Kitchen (4,015 messages), Michelle Lokay (2,489 messages), Richard Sanders (1,188 messages), and William Williams III (2,769 messages). All of these messages are in the same format as the messages found in the Enron corpus. They are missing a couple email headers (e.g., "Received"), but they have the most common headers (e.g., "Subject", "To", "From", etc.).

These collections are all potentially biased for the same two reasons. First, each of the collections is too small to be a representative sample. Second, each collection focuses on only one individual's email; thus, each collection is potentially biased towards the content of a single individual.

3.1.2.3 Graduate Student Legitimate Corpora

The graduate student legitimate corpora consist of 513, 3,407, and 5,287 manually classified legitimate messages that were obtained from three graduate students' personal email accounts (grad1, grad2, and grad3). All of these messages are in a traditional email format, including their original headers.

These corpora are potentially biased for the same two reasons as the Enron personal legitimate corpora. First, each collection of messages is too small to be a representative sample of legitimate messages. Second, each collection is potentially biased towards one individual's email content, making it representative of only one user's email.

3.1.2.4 SpamAssassin Legitimate Corpus

The SpamAssassin corpora contain a legitimate corpus of 6,951 legitimate messages that were collected from various SpamAssassin users. These legitimate messages are in a traditional email format, including their original headers. The messages are also broken into two groups: easy ham (6,451 messages) and hard ham (500 messages). The easy ham messages are "easy" to distinguish from spam, but the hard ham messages are "hard" to distinguish. We omitted the hard ham messages from our evaluation because when the filter was trained with this corpus, it generated extremely erratic classification results. Concretely, the hard ham group is one of the smallest legitimate corpora, and its messages have many spam-like characteristics. As a result, a classifier that is trained with this corpus as its only legitimate sample generates results that show an extremely high degree of variance. The resulting classifier has a very narrow view of legitimate messages and performs worse than the other classifiers that were trained with other small, narrow corpora. For these reasons, using the hard ham messages as a legitimate sample paints an even grimmer picture for the usage of small, narrow corpora in evaluations.

This corpus consists of messages from a number of unique users, and it contains more messages than the other small legitimate corpora. However, it is still potentially biased due to its size. A few thousand messages is not enough to be considered a representative sample of spam messages.

3.1.3 Large Spam Corpus

Our large spam corpus contains over 600K spam messages that were obtained from the publicly available spam corpora maintained by SpamArchive [139]. All of these messages are in a traditional email format, including their original headers.

This corpus does not possess the same potentially biased characteristics that are found in the small corpora described above in Section 3.1.1. It contains more than two orders of magnitude more messages than any of the small spam corpora, and it contains spam messages that were submitted by thousands of unique users. The SpamArchive has been chosen as a representative large corpus in our experiments due to its size and diversity of sources. The experimental results in this chapter will show that such a large corpus produces quantitatively more consistent results than those obtained with the small corpora described in the previous section.

We should observe the limitations of any corpus in the evaluation of email spam filters, which are similar to the limitations of any benchmark (including those used routinely in machine learning and natural language research). The question of whether any fixed benchmark is representative of an application space cannot be answered in a statistical sense when the application space is open and evolving. Nevertheless, well designed benchmarks that represent useful subspaces of the application area provide significant practical benefits in the evaluation and comparison of computer systems and applications. The email spam space is clearly a continuously developing area since new email spam is constantly created (more than 4 billion spam messages per day as counted by MAAWG) by the spammers. Consequently, no fixed corpus can be shown to be "representative" of the email spam space in a statistical sense. Nevertheless, we believe that the SpamArchive corpus covers the email spam area better than other currently published corpora due to its size and diversity

of sources.

3.1.4 Large Legitimate Corpus

Our large legitimate corpus contains 475K messages that were taken from the Enron corpus. In our initial experiments with the published version of the Enron corpus, we discovered a small percentage of messages in the corpus that appeared to be spam. We used a Naïve Bayes-based spam filter, trained with 5K randomly selected spam messages (taken from our SpamArchive spam corpus) and 5K randomly selected legitimate messages (taken from a collection of 700K legitimate messages that were collected privately from newsgroups, personal email, and other user-submitted email messages), to classify the Enron messages. Since statistical filters are generally very effective in distinguishing spam from legitimate emails in static corpora, this procedure was able to find most of the spam messages in the Enron corpus. The messages that the filter labeled as spam were discarded, and then, the remaining messages were manually sampled to ensure their legitimacy. After this process was complete, we were left with 475K Enron messages.

Similar to our large spam corpus (SpamArchive), the Enron corpus does not exhibit the potentially biased characteristics that are evident in the small corpora described above in Section 3.1.2. It contains almost two orders of magnitude more messages than any of the small legitimate corpora, and it contains legitimate messages from more than 150 different users. Similar to our previous discussion about the usefulness of SpamArchive as a benchmark, we consider the Enron corpus as a useful benchmark for legitimate email in an open and evolving application space. Although no fixed corpus can represent the open space of legitimate email in a statistical sense, the Enron corpus is currently the largest (and most diverse) published email corpus without privacy problems (due to the circumstances that led to its publication during the Enron trial).

3.2 Evaluation of Small Corpora Training Sets

In this section, we present a comparative study of spam filtering experiments using representative large corpora and small corpora. The purpose of this experimental evaluation is to test the following hypothesis:

Hypothesis: A training set taken from a small corpus is likely to introduce bias into the trained filter.

As mentioned in Section 3.1, we use the term *bias* to refer to a significant variation in filter performance when we vary the content of the test set. This is a problem when we expect reliable and reproducible filter performance over a wide range of messages in the test set. An intuitive explanation of our hypothesis is that we assume a comprehensive and representative training set is needed for reproducible filter performance. Using small corpora to create training sets only covers a small subset of the features in the set of known spam and legitimate email messages. Consequently, when the test messages contain features that are unknown to the filter, the filter's performance becomes less reproducible.

In Section 3.2.1, we describe our experimental setup and the experiments we performed. In Section 3.2.2, we summarize the experimental results for the training sets taken from small corpora, showing the bias introduced into a Naïve Bayesian filter. In Section 3.3, we evaluate the performance of the filter when it is trained with large corpora.

3.2.1 Methodology

To evaluate the biasing influence of each of our small corpora, we conducted a number of experiments in which we trained our spam filter with numerous training set combinations and observed its filtering performance on a test set taken from large corpora. In each of these experiments, the filter's training set was determined by consulting the training matrix provided in Table 2, and the test set was taken from the large spam and large legitimate corpora described in Sections 3.1.3 and 3.1.4. Using the training matrix, four groups of experiments emerged: (1) training with small spam and small legitimate corpora, (2) training with large spam and small legitimate corpora, (3) training with small spam and large legitimate corpora, and (4) training with large spam and large legitimate corpora.

To create a fair comparison, we used 481 spam messages and 481 legitimate messages in each of the training sets for all four experimental groups. We chose 481 messages because it is the largest common corpus size for the corpora described above in Section 3.1. For

Table 2: Four combinations of training sets.

		Legitimate: Large	Legitimate: Small
Spam	Large	Enron SpamArchive	L-s Legit., beck-s, farmer-d, kaminski-v, kitchen-l, lokay-m, sanders-r, williams-w3, grad1, grad2, grad3, SA Legit. SpamArchive
	Small	Enron L-s Spam SA Spam	L-s Legit., beck-s, farmer-d, kaminski-v, kitchen-l, lokay-m, sanders-r, williams-w3, grad1, grad2, grad3, SA Legit. L-s Spam SA Spam

the corpora with more than 481 messages (i.e., all of the corpora except the Ling-spam spam corpus), we randomly selected a 481 message subset. Additionally, to accommodate the varying message formats of each corpus, the "Subject" header was the only header information we used in our evaluation. Unfortunately, this header represents the only header information that is shared by all of the corpora, and in the interest of fair comparisons, we were forced to omit all other header information. Also, due to the ease with which spammers can spoof headers, it is expected that they will manipulate the headers with increasing sophistication to hide any unintended traces of spam.

Once the spam and legitimate training sets were selected, we used them to train our Naïve Bayesian filter. For our Naïve Bayesian learning spam filter, we chose the Weka software package [152]. Weka is an open source collection of machine learning algorithms

that has become a standard implementation tool in the machine learning community. The Weka Naïve Bayes implementation we chose (weka.classifiers.bayes.NaiveBayes) uses the multi-variate Bernoulli model [106, 134], which is the same event model used by most previous Naïve Bayesian filter evaluations.

Then, we used 1K spam messages and 1K legitimate messages as a test set to evaluate the trained filter's classifying performance. The messages for this test set were randomly selected from our large corpora (i.e., the SpamArchive corpus for the spam messages and our Enron corpus for the legitimate messages). This training and classification process was performed ten times, using ten unique seed values for the random message selection.

In the first group of experiments, the filter was trained with small spam and small legitimate corpora. Since we evaluated 2 small spam corpora and 12 small legitimate corpora, this group of experiments resulted in 24 different training set configurations. The second group of experiments involved training the filter with large spam and small legitimate corpora. Using our large spam corpus and the 12 small legitimate corpora, this group of experiments used 12 different training set configurations. In the third group of experiments, the filter was trained with small spam and large legitimate corpora. We used two small spam corpora and one large legitimate corpus in our evaluation; thus, this group of experiments had 2 different training set configurations. The final group of experiments trained the filter with large spam and large legitimate corpora. This group of experiments only had one training configuration because we evaluated one large spam corpus and one large legitimate corpus.

3.2.2 Experimental Results

To evaluate our hypothesis about the biasing effect of small corpora, we present our experimental results for the four experiment groups described above in Section 3.2.1. To determine our filter's performance when it is trained with the various training sets, we evaluate the filter's false positive and false negative rates. Additionally, in each of our experiments, we varied the number of retained features that were used to train the filter between 80 and 10K (80, 160, 320, 640, 1K, 2K, ..., 10K). These retained features were selected using the Information Gain feature selection algorithm described in Chapter 2. Thus, in most of the

figures in this chapter, we present the filter's false positive or false negative results as a function of the number of retained features. The equations for the filter's false positive and false negative rates are the following:

$$false\ positive\ rate = 1 - \frac{n_{L\to L}}{N_L}$$

and

$$false\ negative\ rate = 1 - \frac{n_{S\to S}}{N_S}.$$

In the above equations, $n_{L\to L}$ is the number of legitimate test set messages the filter correctly identified as legitimate, N_L is the total number of legitimate test set messages, $n_{S\to S}$ is the number of spam test set messages the filter correctly identified as spam, and N_S is the total number of spam test set messages.

3.2.2.1 Training with Small Spam and Small Legitimate Corpora

Our first set of experiments tested our hypothesis by exclusively using the small, potentially biased corpora for training. The reason each of these corpora is potentially biased is given above in Section 3.1. We note that Figure 1 and Table 3 are designed to compare the false positive rates of several distinctly trained filters with each other. Similarly, Figure 2 and Table 4 compare the false negative rates of those filters. This presentation is not intended for studying the effectiveness of any individual filter or method.

In Figure 1, we show the false positive rates of our filter when it was trained with the Ling-spam spam corpus and each of the small legitimate corpora. Each line shows the average result of a set of 10 experimental runs. The y-axis of this figure corresponds to the filter's false positive rate, and the x-axis corresponds to the number of retained features that were used for training the filter. In Table 3, we present the filter's *stable* false positive results. These results correspond to the filter's performance in Figure 1 when the lines stop fluctuating. Three values are given: the false positive rate's average, standard deviation, and coefficient of variance. We use these common statistical measures to show the existence of variability in filter performance. Due to their pervasiveness and magnitude, we believe the variability will not disappear with other statistical assumptions (e.g., distribution different

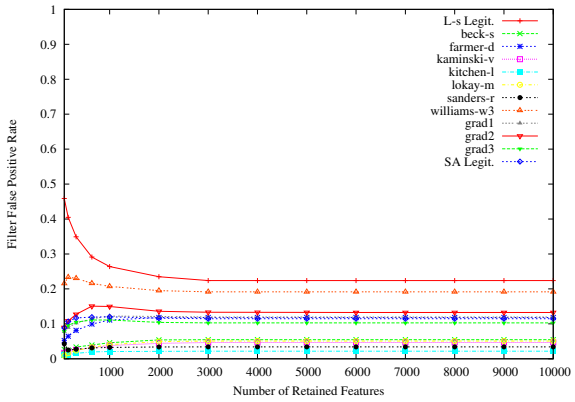


Figure 1: False positive rates for the Ling-spam spam and small legitimate training sets.

Table 3: Stable false positive results for the Ling-spam spam and small legitimate training sets.

Legit. Corpus	Spam Corpus	False Pos. Avg.	False Pos. Std. Dev.	False Pos. C.o.V.
L-s Legit.	L-s Spam	0.224	0.026	11.5%
beck-s		0.055	0.008	15.3%
farmer-d		0.118	0.017	14.0%
kaminski-v		0.047	0.007	15.5%
kitchen-l		0.022	0.007	32.5%
lokay-m		0.051	0.008	16.6%
sanders-r		0.034	0.006	17.5%
williams-w3		0.192	0.015	7.94%
grad1		0.119	0.010	8.53%
grad2		0.133	0.012	8.97%
grad3		0.103	0.013	12.6%
SA Legit.		0.115	0.008	7.24%

than normal). It is not our intention to study the detailed statistical nature of these variations in filter performance in this particular set of corpora.

Figure 1 clearly shows a wide variation in the false positive rates produced by the filter when it was trained with the various small corpora. On the other hand, Table 3 displays consistent behavior within the results for each individual training set. Both the standard deviations and the coefficients of variance are small. Thus, these findings illustrate a contrast between the consistent behavior of a single trained filter and the inconsistent results observed when comparing all of the trained filters. As a result, we observe that training sets taken from small corpora can produce filters with very different, individually consistent, behaviors.

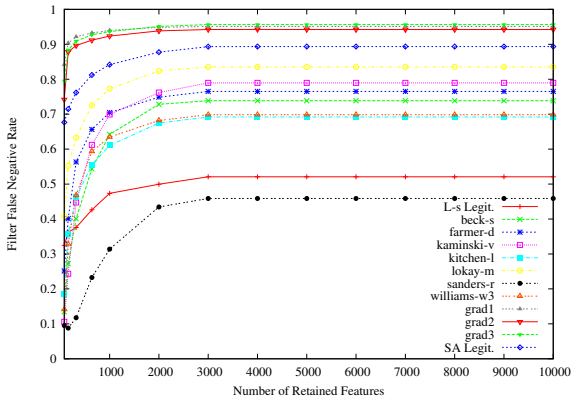


Figure 2: False negative rates for the Ling-spam spam and small legitimate training sets.

Table 4: Stable false negative results for the Ling-spam spam and small legitimate training sets.

Legit. Corpus	Spam Corpus	False Neg. Avg.	False Neg. Std. Dev.	False Neg. C.o.V.
L-s Legit.	L-s Spam	0.521	0.051	9.78%
beck-s		0.739	0.019	2.55%
farmer-d		0.765	0.018	2.38%
kaminski-v		0.790	0.017	2.09%
kitchen-l		0.692	0.050	7.17%
lokay-m		0.835	0.011	1.30%
sanders-r		0.459	0.064	14.0%
williams-w3		0.698	0.024	3.51%
grad1		0.951	0.009	0.92%
grad2		0.942	0.010	1.05%
grad3		0.957	0.007	0.70%
SA Legit.		0.894	0.012	1.33%

Additionally, we observe that the variation among the twelve small training sets in Figure 1 is quite pronounced with three distinct trends. At the bottom of the figure, the five lines primarily below 5% false positives show a gradual increase in false positive rates as the number of retained features grows. These five lines correspond to five of the seven Enron personal legitimate corpora (beck-s, kaminski-v, kitchen-l, lokay-m, and sanders-r). At the top of the figure, the two lines above 15% false positives show a gradual decrease in false positive rates as the number of retained features grows. These two lines correspond to the Ling-spam legitimate corpus and one of the Enron personal legitimate corpora (williams-w3). In the middle of the figure, the five lines between 5% and 15% false positives show a sharp initial increase in false positive rates (between 80 and 1K retained

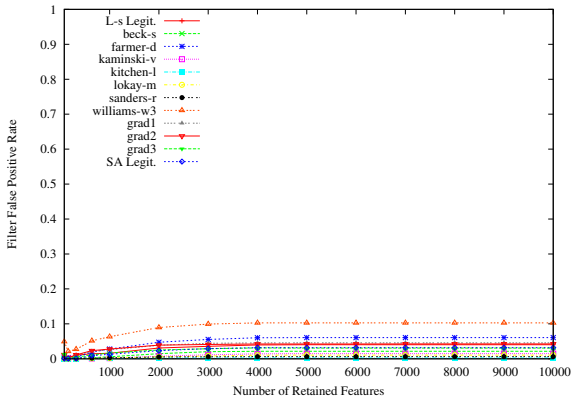


Figure 3: False positive rates for the SpamAssassin spam and small legitimate training sets.

Table 5: Stable false positive results for the SpamAssassin spam and small legitimate training sets.

Legit. Corpus	Spam Corpus	False Pos. Avg.	False Pos. Std. Dev.	False Pos. C.o.V.
L-s Legit.	SA Spam	0.040	0.018	44.0%
beck-s		0.022	0.004	20.5%
farmer-d		0.061	0.012	19.2%
kaminski-v		0.015	0.004	26.6%
kitchen-l		0.003	0.002	60.4%
lokay-m		0.011	0.003	32.1%
sanders-r		0.006	0.003	42.6%
williams-w3		0.103	0.020	19.2%
grad1		0.046	0.007	14.3%
grad2		0.043	0.010	22.8%
grad3		0.031	0.005	15.0%
SA Legit.		0.032	0.007	23.2%

features), followed by a slight decrease in false positive rates. These five lines correspond to one of the Enron personal legitimate corpora (farmer-d), the three graduate student legitimate corpora (grad1, grad2, and grad3), and the SpamAssassin legitimate corpus. As our goal is simply to show the existence of a bias in the filter due to small corpora, we do not attempt to explain any individual curve further.

Figure 2 and Table 4 describe the false negative performance for the filters in Figure 1 and Table 3. In Figure 2, all of the lines show a monotonically increasing false negative rate as the number of retained features grows, and we observe a very wide range of false negative values. For 80 retained features, the false negative variation stretches from less than 10% to more than 90%. For the stable values between 3K and 10K retained features, the false

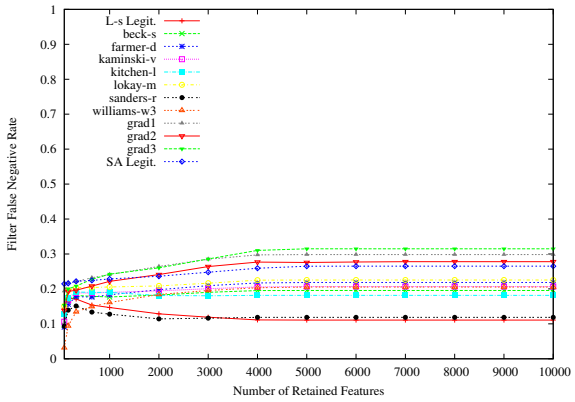


Figure 4: False negative rates for the SpamAssassin spam and small legitimate training sets.

Table 6: Stable false negative results for the SpamAssassin spam and small legitimate training sets.

Legit. Corpus	Spam Corpus	False Neg. Avg.	False Neg. Std. Dev.	False Neg. C.o.V.
L-s Legit.	SA Spam	0.111	0.016	14.3%
beck-s		0.195	0.016	8.06%
farmer-d		0.218	0.016	7.46%
kaminski-v		0.207	0.015	7.35%
kitchen-l		0.182	0.011	6.05%
lokay-m		0.226	0.018	8.11%
sanders-r		0.119	0.026	22.0%
williams-w3		0.205	0.016	7.88%
grad1		0.298	0.030	9.94%
grad2		0.278	0.036	13.0%
grad3		0.315	0.029	9.07%
SA Legit.		0.265	0.020	7.63%

negatives range from 46% to 96%. Thus, both Figure 1 and Figure 2 show the wide variation that supports our hypothesis about the biasing effect of small corpora. Additionally, Table 4 shows that the filter exhibited consistent behavior for each individual small training set (i.e., the standard deviations and coefficients of variance are small), giving even more evidence that small, biased training sets can produce filters with very different, individually consistent, behaviors.

To help gauge the influence of the Ling-spam spam corpus on the experimental results, we ran similar experiments using the SpamAssassin spam corpus. Figure 3 shows the false positive rates for the filter when it was trained with the SpamAssassin spam corpus and each of the small legitimate corpora. Table 5 displays the filter's stable false positive results.

The results in Figure 3 display a smaller range in the filters' variation compared to those found in the Ling-spam experiments. The filter with the worst performance only had a false positive rate around 10%, and all of the training sets represented in this figure resulted in a single trend (i.e., increasing with the number of retained features). However, we still observe variation in the form of vertical separation between each of the lines. Additionally, in Table 5, we observe that each filter's performance is consistent (i.e., the standard deviations and coefficient of variances are small) for a given training set. Thus, as with the Ling-spam experiments above, these results indicate that training sets taken from small corpora generate filters that exhibit diverse, yet individually consistent, performances.

Figure 4 and Table 6 show the false negative results for the same experiments as Figure 3 and Table 5. Instead of the wide variation seen in Figure 2, this figure displays a maximum false negative rate of less than 32%. However, three distinct trends emerge. At the bottom of the figure, two monotonically decreasing lines stay primarily below 15% false negatives. These two lines correspond to the Ling-spam legitimate corpus and one of the Enron personal legitimate corpora (sanders-r). In the middle of the figure, the six primarily flat lines stay around 20% false negatives. These lines correspond to six of the seven Enron personal legitimate corpora (beck-s, farmer-d, kaminski-v, kitchen-l, lokay-m, williams-w3). At the top of the figure, four monotonically increasing lines grow quickly to above 20% false negatives. These lines correspond to the three graduate student legitimate corpora (grad1, grad2, and grad3) and the SpamAssassin legitimate corpus.

Although the vertical dispersion of these lines is much less than that found in Figure 2, the filters' performance showcased in this figure is still very inconsistent. The lack of uniformity across the various training sets in these figures supports our original hypothesis for the case where both the spam and legitimate training sets were taken from small corpora. On the other hand, Table 6 shows once again that each individual training set results in consistent performance. Thus, even though the training sets generate comparatively diverse results, their individual results are consistent.

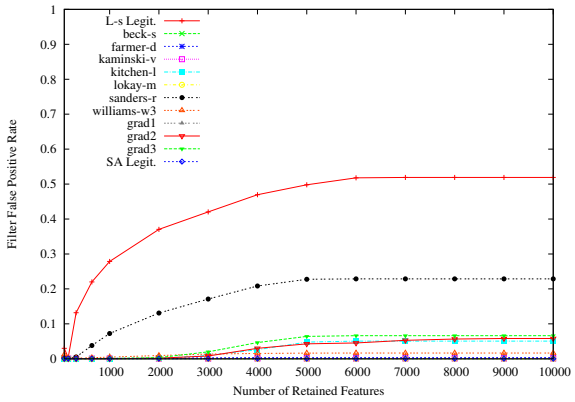


Figure 5: False positive rates for the SpamArchive spam and small legitimate training sets.

Table 7: Stable false positive results for the SpamArchive spam and small legitimate training sets.

Legit. Corpus	Spam Corpus	False Pos. Avg.	False Pos. Std. Dev.	False Pos. C.o.V.
L-s Legit.	SpamArchive	0.5190	0.0338	6.508%
beck-s		0.0004	0.0005	129.1%
farmer-d		0.0036	0.0039	108.1%
kaminski-v		0.0010	0.0016	163.3%
kitchen-l		0.0507	0.0650	128.2%
lokay-m		0.0001	0.0003	316.2%
sanders-r		0.2287	0.0505	22.10%
williams-w3		0.0164	0.0104	63.32%
grad1		0.0008	0.0008	98.60%
grad2		0.0582	0.0342	58.75%
grad3		0.0662	0.0481	72.72%
SA Legit.		0.0002	0.0004	210.8%

3.2.2.2 Training with Large Spam and Small Legitimate Corpora

With the wide variation found in Figures 1 through 4, the next question is whether the amount of bias in the training set is linked to the variation in filter performance. Our next set of experiments is similar to those in Section 3.2.2.1; however, in this set we reduced the amount of bias during training by using spam training sets that were produced by random selection from large corpora. The legitimate training sets were the same as in the previous set of experiments, and the test set was also the same.

Figure 5 shows the false positive rates when the filter was trained using the SpamArchive corpus and the small legitimate corpora, and Table 7 presents the filter's false positive

results when 10K features were retained (i.e., when the results were stable). Compared to Figures 1 and 3, Figure 5 displays more uniformity for the most part. In fact, 9 out of 12 lines are almost identical between 80 and 2K features. However, since the figure shows two outlying curves with more than 20% and 50% maximum false positives, the results still show significant variation. These two outlying lines correspond to the Ling-spam legitimate corpus and one of the Enron personal legitimate corpora (sanders-r). Additionally, a spread exists between 0% and 7% false positives among the other ten lines when the number of retained features is greater than 2K.

Table 7 is harder to interpret for these results because the average performance of most of the filters is so good (i.e., the false positive rates are small). In many cases, the standard deviations are as large as the averages; however, this does not indicate inconsistencies for a given training set. For those particular training sets, the filter performs so well that a single misclassification is enough to skew the results. For example, in this experiment, a filter trained with the SpamArchive corpus and the kaminski-v corpus only misclassified an average of 1 out of 1K legitimate messages in the test set as spam. Thus, if the filter misclassified 2 out of 1K legitimate messages in one of the experimental runs, it would be enough to greatly influence the standard deviation results for that training set. With this observation in mind, it is safe to conclude that each individual filter's results were consistent, despite the variation among the various filters.

Figure 6 and Table 8 show the false negative results for the same experiments as Figure 5 and Table 7. Compared to Figures 2 and 4, Figure 6 displays a much smaller variation (less than a 25% maximum false negative rate) and a single downward trend. However, at the stable false negative rates above 3K retained features, the lines are still approximately uniformly spread (in terms of vertical distance) between about 2% and 14%. Additionally, similar to Tables 4 and 6, Table 8 shows that the filter exhibited consistent behavior for each individual small training set (i.e., the standard deviations and coefficients of variance are small).

Both Figures 5 and 6 show that replacing the spam training sets taken from small corpora with spam training sets taken from large corpora reduced the variation in the results, which

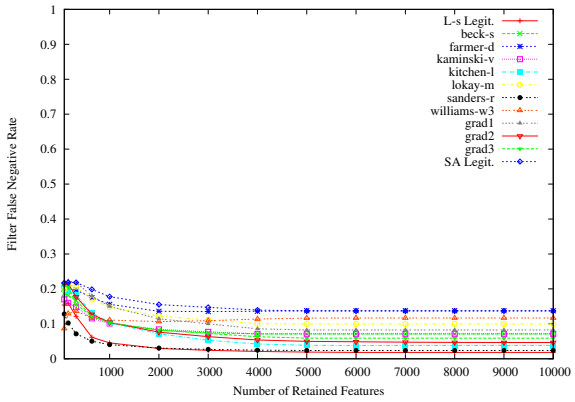


Figure 6: False negative rates for the SpamArchive spam and small legitimate training sets.

Table 8: Stable false negative results for the SpamArchive spam and small legitimate training sets.

Legit. Corpus	Spam Corpus	False Neg. Avg.	False Neg. Std. Dev.	False Neg. C.o.V.
L-s Legit.	SpamArchive	0.018	0.006	30.6%
beck-s		0.071	0.013	18.1%
farmer-d		0.138	0.019	14.1%
kaminski-v		0.071	0.018	25.1%
kitchen-l		0.038	0.007	18.8%
lokay-m		0.099	0.015	14.7%
sanders-r		0.024	0.005	21.1%
williams-w3		0.117	0.016	13.4%
grad1		0.083	0.013	15.9%
grad2		0.047	0.012	24.7%
grad3		0.059	0.016	27.9%
SA Legit.		0.137	0.015	11.2%

is consistent with our original hypothesis. In the next section, we investigate the effect of only replacing the legitimate training sets taken from small corpora with legitimate training sets taken from large corpora.

3.2.2.3 Training with Small Spam and Large Legitimate Corpora

With the reduction in variation observed in Section 3.2.2.2, a plausible question is whether replacing only the legitimate training sets taken from small corpora with legitimate training sets taken from large corpora would have a similar effect as replacing only the spam training sets.

Figure 7 shows the false positive rates for our filter when it was trained with each of

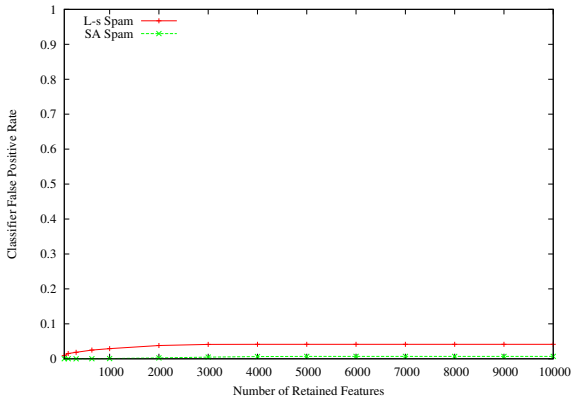


Figure 7: False positive rates for the small spam and Enron legitimate training sets.

Table 9: Stable false positive results for the small spam and Enron legitimate training sets.

Legit. Corpus	Spam Corpus	False Pos. Avg.	False Pos. Std. Dev.	False Pos. C.o.V.
Enron	L-s Spam	0.042	0.007	17.9%
	SA Spam	0.007	0.003	44.0%

the small spam corpora and our Enron corpus, and Table 9 displays the filter's stable false positive results. These results are very similar to the best results found in the corresponding experiments in Section 3.2.2.1. The L-s Spam results in Figure 7 and Table 9 are better than all but two (kitchen-l and sanders-r) of the results in Figure 1 and Table 3, and the SA Spam results are better than all but two (kitchen-l and sanders-r) of the results in Figure 3 and Table 5. Thus, replacing the legitimate training sets taken from small corpora with legitimate training sets taken from large corpora reduced the bias in the false positive results.

Figure 8 and Table 10 show the false negative rates for the same experiments as Figure 7 and Table 9. Unlike the reduction found in false positive results, the two lines in Figure 8 show a wide variation (from 20% for SA Spam to 80% for L-s Spam). Additionally, these results are slightly higher than the average of the 12 results found in the corresponding experiments in Section 3.2.2.1. Although we only have two small spam training sets in this experiment (L-s Spam and SA Spam), the mixed results reaffirm our hypothesis.

In summary, sections 3.2.2.1, 3.2.2.2, and 3.2.2.3 show that our hypothesis holds for all

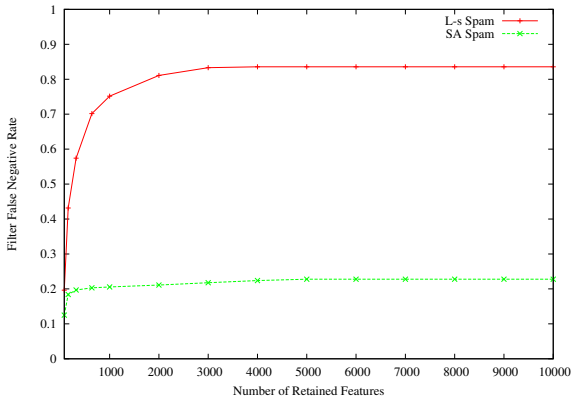


Figure 8: False negative rates for the small spam and Enron legitimate training sets.

Table 10: Stable false negative results for the small spam and Enron legitimate training sets.

Legit. Corpus	Spam Corpus	False Neg. Avg.	False Neg. Std. Dev.	False Neg. C.o.V.
Enron	L-s Spam	0.836	0.017	1.99%
	SA Spam	0.228	0.019	8.22%

three cases in Table 2 that use small corpora for their training sets. In the next section, we evaluate the filter training when using large corpora, as a comparison to training filters with small corpora.

3.3 Evaluation of Large Corpora Training Sets

In this section, we continue our comparative study of spam filtering experiments. First, in Section 3.3.1, we present the performance of our filter when it was trained with large spam and large legitimate corpora. Then, in Section 3.3.2, we experimentally evaluate the appropriate size for large corpora in spam filtering research.

3.3.1 Training with Large Spam and Large Legitimate Corpora

In each of the three previous training set combinations, we witnessed some degree of bias. Thus, the remaining alternative in Table 2 is using training sets taken from large spam and large legitimate corpora to train our spam filter. We note that the converse of our hypothesis (i.e., large training sets will not introduce bias into filters) may not be strictly

Table 11: Stable false positive results for the SpamArchive spam and Enron legitimate training sets.

Legit. Corpus	Spam Corpus	False Pos. Avg.	False Pos. Std. Dev.	False Pos. C.o.V.
Enron	SpamArchive	0.0001	0.0003	316.2%

true. The intuitive explanation for this situation is that the reliable performance of our filter depends on the comprehensiveness of the feature coverage in the set of test messages. While large corpora increase the coverage, there may be some pathological cases where even large corpora fail for a particular training set/test set combination. Nevertheless, when both the training set and the test set are extracted from the same corpus (large or small), the feature coverage tends to be representative. Additionally, we will show that when large corpora are used to train the filters, the resulting filter is able to handle a variety of test set combinations in a reliable manner.

Table 11 displays the stable false positive results for the experiments using the SpamArchive corpus and our Enron corpus. From these results, we can observe that the use of training sets taken from large spam and large legitimate corpora resulted in very low, consistent false positive results. In fact, these false positive rates are lower than all of the previous experiments, reinforcing our original hypothesis that small, biased training sets introduce a bias into the filter's performance.

Figure 9 and Table 12 show the false negative results for the same experiments as Table 11. These false negative values exhibit a downward trend, and they are consistently lower than those found in Section 3.2.2.3 and comparable to those found in Section 3.2.2.2.

3.3.2 How Large Should the Corpora Be?

Previously, we showed that a training set taken from small corpora introduces a bias into the performance of a learning spam filter. In this section, we examine the effect of varying the training and test set sizes on the performance of the same spam filter to find the the appropriate sample sizes for evaluations using large corpora. To perform this examination, we exclusively used our large corpora (i.e., our SpamArchive corpus for spam messages and

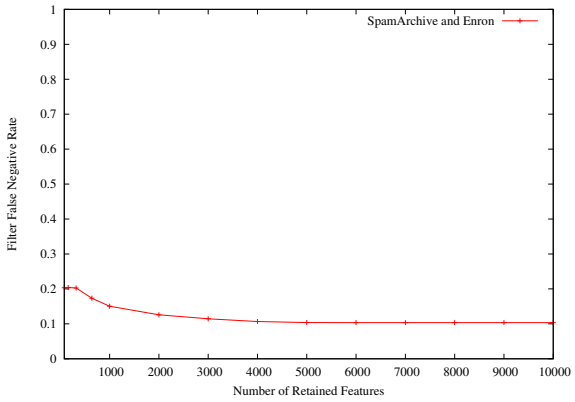


Figure 9: False negative rates for the SpamArchive spam and Enron legitimate training sets.

Table 12: Stable false negative results for the SpamArchive spam and Enron legitimate training sets.

Legit. Corpus	Spam Corpus	False Neg. Avg.	False Neg. Std. Dev.	False Neg. C.o.V.
Enron	SpamArchive	0.1035	0.0176	17.02%

our Enron corpus for legitimate messages) for the training and test sets, and we retained 10K features using the Information Gain feature selection algorithm. We used 6 different training set sizes (500, 1K, 5K, 10K, 50K, and 100K) and 5 different test set sizes (500, 1K, 5K, 10K, and 50K) for a total of 30 different training and test set combinations.

Table 13 presents the filter's false positive results for the various training and test set combinations. Three values are given: the false positive rate's average, standard deviation, and coefficient of variance. We omit a graphical representation of these results because the false positive rates are all essentially zero. However, we still observe a couple interesting trends from the table. First, for a given training set, the results for the various test sets are very similar, indicating a strong degree of internal consistency for our large corpora. Second, the use of smaller test sets can result in deceptive results. For example, all of the training set sizes generated a filter that was able to correctly classify all of the legitimate messages in the 500 and 1K message test sets. However, for larger test sets, these filters were not able to maintain this level of excellence. Their performance was still extremely

Table 13: False positive results for various training and test set sizes.

Training Set Size	Test Set Size	False Pos. Avg.	False Pos. Std. Dev.	False Pos. C.o.V.
500	500	0	0	0%
	1K	0	0	0%
	5K	2.00e-4	2.83e-4	141%
	10K	1.60e-4	1.58e-4	98.6%
	50K	1.40e-4	8.06e-5	57.5%
1K	500	0	0	0%
	1K	0	0	0%
	5K	8.00e-5	1.69e-4	211%
	10K	6.00e-5	9.66e-5	161%
	50K	8.00e-5	9.43e-5	118%
5K	500	0	0	0%
	1K	0	0	0%
	5K	8.00e-5	1.69e-4	211%
	10K	4.00e-5	8.43e-5	211%
	50K	4.00e-5	3.77e-5	94.3%
10K	500	0	0	0%
	1K	0	0	0%
	5K	4.00e-5	1.26e-4	316%
	10K	6.00e-5	1.35e-4	225%
	50K	2.00e-5	2.83e-5	141%
50K	500	0	0	0%
	1K	0	0	0%
	5K	0	0	0%
	10K	2.00e-5	6.32e-5	316%
	50K	2.40e-5	2.80e-5	117%
100K	500	0	0	0%
	1K	0	0	0%
	5K	4.00e-5	1.26e-4	316%
	10K	2.00e-5	6.32e-5	316%
	50K	4.80e-5	4.38e-5	91.3%

good; however, it was not as good as one would expect based on the performance with the smaller test sets.

The variation of the results for larger test sets appears to be high (i.e., the coefficient of variances are large); however, this is vastly overstated. Since the average performance of the filter is so high (i.e., the false positive rate is essentially zero), a misclassification of one message in one of the 10 experimental runs is enough to generate the observed variation. Additionally, it is important to note that this variation decreases as the test set size increases. Thus, the use of larger test sets is important for generating consistent, reproducible results.

Figure 10 shows the false negative results for the various training and test set configurations. This figure presents a couple important trends. First, as the training set increases, the filter's false negative rate decreases. This trend clearly shows the importance of using

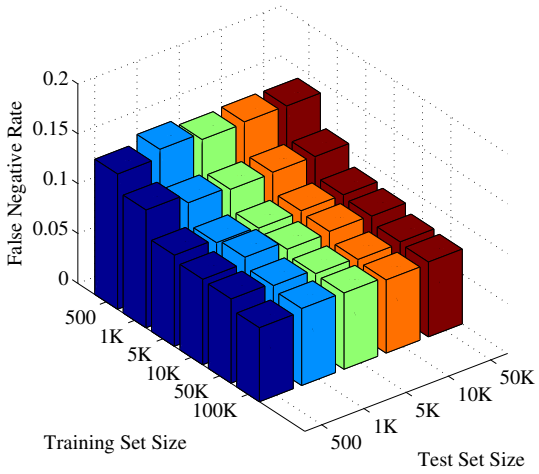


Figure 10: False negatives rates for varying training and test set sizes.

larger training sets, and it reiterates our previous conclusion about the biasing effect of small corpora. If the filters are not trained with a sufficiently large number of messages, their true performance will not be known. Second, as shown above in the false positive results, the false negative rates remain about the same as the test set size increases. However, one caveat of this observation is that very small test set sizes (e.g., 500 messages) are not very reliable indicators of the filters' performance. In some cases, these small test sets overestimate the performance of the filters, and in other cases, they underestimate the filters' performance. Thus, the larger test sets are necessary for the most accurate evaluation of the filters' performance.

Table 14 presents the false negative rate averages and variation results for the various training and test set configurations when 10K features were retained. These results show two very interesting trends. First, as the test set size increases, the variation of the results decreases. This result reaffirms our claim that using larger test sets results in more reliable and more reproducible results. Second, for larger test sets (e.g., larger than 1K), we observe a generally decreasing trend for the variation of the results as the training set size increases. This decrease is particularly noticeable when the training set size increases from 1K to 5K. As a result, we can conclude that using large training and test set sizes produces more reliable performance results. Thus, we have provided even more evidence that large

Table 14: False negative results for various training and test set sizes.

Training Set Size	Test Set Size	False Neg. Avg.	False Neg. Std. Dev.	False Neg. C.o.V.
500	500	0.136	2.73e-2	20.1%
	1K	0.144	2.71e-2	18.8%
	5K	0.138	2.75e-2	19.9%
	10K	0.139	2.41e-2	17.3%
	50K	0.139	2.45e-2	17.6%
1K	500	0.118	1.24e-2	10.5%
	1K	0.109	2.05e-2	18.8%
	5K	0.106	1.75e-2	16.5%
	10K	0.107	1.76e-2	16.4%
	50K	0.107	1.70e-2	16.0%
5K	500	9.12e-2	3.15e-2	34.6%
	1K	8.76e-2	1.44e-2	16.5%
	5K	8.60e-2	5.45e-3	6.34%
	10K	8.70e-2	3.46e-3	3.98%
	50K	8.67e-2	3.62e-3	4.17%
10K	500	8.56e-2	1.31e-2	15.3%
	1K	9.16e-2	1.87e-2	20.4%
	5K	8.33e-2	4.92e-3	5.90%
	10K	8.50e-2	3.92e-3	4.61%
	50K	8.40e-2	2.18e-3	2.59%
50K	500	8.44e-2	1.45e-2	17.2%
	1K	8.16e-2	1.60e-2	19.7%
	5K	7.73e-2	4.54e-3	5.87%
	10K	7.52e-2	3.13e-3	4.16%
	50K	7.67e-2	2.15e-3	2.80%
100K	500	7.40e-2	1.48e-2	19.9%
	1K	7.70e-2	1.51e-2	19.6%
	5K	7.63e-2	6.28e-3	8.23%
	10K	7.32e-2	2.14e-3	2.92%
	50K	7.50e-2	2.17e-3	2.90%

corpora are vital for producing reliable and reproducible results during learning spam filter evaluations.

3.4 Related Work

Many experimental evaluations have been performed using statistical spam filters. Pantel and Lin [117] and Sahami et al. [127] were the first researchers to apply machine learning techniques to the spam filtering problem. Pantel and Lin showed that a Naïve Bayes algorithm could outperform RIPPER [35] - a "keyword-spotting" rule learner developed by W. W. Cohen. Sahami et al. explored the effectiveness of using a Naïve Bayes algorithm for spam filtering and found that using manually derived heuristics improved the algorithms's performance. However, both evaluations used small, personally biased spam and legitimate corpora that are unpublished. Drucker et al. [46] evaluated the use of support vector

machines (SVMs) as a spam filtering solution, comparing it to RIPPER, Rocchio [91, 133], and a boosting algorithm that used classification trees built using a version of C4.5 [123]. The result of this work was that SVMs and the boosting algorithm exhibit similar error rates that are better than those obtained with Rocchio. Unfortunately, this evaluation relied upon two small, biased email data sets that are unpublished.

Androutsopoulos et al. [6] extended previous spam filtering work by creating a publicly available email corpus (Ling-spam) and using it to analyze the effect of attribute-set size, training-corpus size, lemmatization, and stop-lists on the performance of a filter based on the Naïve Bayes algorithm. Then, they took this work a step further [7], introducing another publicly available email corpus (PU1) and using it to perform a similar analysis of their Naïve Bayesian filter's performance. They also confirmed previous results that showed a Naïve Bayesian filter outperformed a keyword-based filter. Androutsopoulos et al. [8] used the Ling-spam corpus to compare a Naïve Bayesian filter and a memory-based algorithm called TiMBL [41]. Their evaluation showed that the Naïve Bayesian filter and TiMBL exhibited similar performance, and both algorithms outperformed a keyword-based filter. Sakkis et al. [128] explored the effectiveness of combining classifiers, using stacked generalization, to filter spam. Using Ling-spam, they found that stacked classifiers performed better than the individual classifiers used in the stacking. All of these evaluations were novel; however, they were all limited by their use of small, biased email corpora (Ling-spam and PU1).

Carreras and Marquez [26] showed that the AdaBoost.MH [132] learning algorithm was more effective than Decision Trees and a Naïve Bayes algorithm when applied to the PU1 email corpus. However, they also questioned the wide-scale applicability of their results due to PU1's small size. Hidalgo [80] used the Ling-spam corpus to compare the performance of filters based on C4.5, Naïve Bayes, PART, Support Vector Machines, and Rocchio. However, the author was not pleased with the current state of email data sets. In fact, he rejected PU1 for the same reason we omitted it from our evaluation – it uses private, encoded messages. Additionally, although Ling-spam was used in the evaluation, the author did not believe Ling-spam was a representative sample of legitimate email messages because it was collected from a linguist newsgroup. Schneider [134] used the Ling-spam and PU1 corpora

to evaluate two different statistical event models: a multi-variate Bernoulli model and a multinomial model. He also experimented with different feature selection algorithms that were derived from Information Gain. Androutsopoulos et al. [9] and Zhang et al. [156] both used small corpora to evaluate the performance of various spam filters. Androutsopoulos et al. used the PU123A corpora to compare filters based on Naïve Bayes, Support Vector Machines, and LogitBoost [61]. Zhang et al. used the Ling-spam, PU1, and SpamAssassin spam corpora to compare filters based on Naïve Bayes, Maximum Entropy Model [18], TiMBL, Support Vector Machines, and AdaBoost [60]. Although all of these evaluations provide interesting results and insight into spam filtering, they are limited by the small corpora they used in their experiments.

Despite the reliance on small corpora in early spam filtering evaluations, some researchers are beginning to incorporate larger corpora into their experiments. An important effort that helped motivate this gradual shift towards using larger corpora was the TREC Spam Track, which gave researchers an opportunity to evaluate filtering approaches on larger corpora. This workshop also generated two publicly available data sets: the TREC 2005 Spam Corpus [39] and the TREC 2006 Spam Corpus [40]. Goodman and Yih [65] used the TREC 2005 Spam corpus to validate a simple discriminative model based on gradient descent of a logistic regression model. Cormack and Bratko [38] used the TREC 2005 Spam corpus, along with the Ling-spam and PU1 data sets, to compare various filtering algorithms (e.g., SVM, logistic regression, etc.), evaluation models (on-line and batch), and message preprocessing approaches (e.g., removing headers, mapping alphanumeric strings to unique integers, etc.). These recent efforts are encouraging because they suggest that other researchers are making similar observations to those illustrated in our work. This gradual shift also serves to further validate our work because we are the first to provide experimental evidence that supports using large corpora in spam filtering experiments.

3.5 Summary

Since spam only became a serious problem in recent years, spam filter evaluation is a relatively new research area. Early spam filtering literature [6, 7, 8, 9, 68, 84, 117, 127, 156]

is dominated by experiments that rely on small corpora. This situation is similar to the early research in the machine learning and natural language processing fields, where small scale experiments preceded the consensus adoption of large corpora [13, 14, 125] as the basis for reliable and reproducible experimental evaluations.

Inspired by lessons learned in machine learning and natural language processing, we conducted a comparative study of learning spam filtering experiments using large corpora (on the order of hundreds of thousands of messages, taken from published email archives) and small corpora (on the order of a few thousand messages, obtained from sources cited in current spam filtering literature). To distinguish spam from legitimate email, the experiments use two training sets, one containing representative samples of spam, and the other containing representative samples of legitimate email. Our study covers the four filter training set combinations: (1) small spam and small legitimate corpora, (2) large spam and small legitimate corpora, (3) small spam and large legitimate corpora, and (4) training with large spam and large legitimate corpora. We studied both the false positive and false negative rates of a Naïve Bayesian learning spam filter, and the results show that small corpora lead to significantly less predictable results when compared to the results obtained by training and testing using large corpora. The filters trained with small corpora produced false positive rates varying from 0% to 46% and false negative rates varying from 3% to 96%; however, the filters trained with large corpora only produced false positive rates consistently around 0% and false negative rates varying from 10% to 20%.

This comparative study is significant for two reasons. Scientifically, it recommends methodological changes in the future experimental evaluations of learning spam filters, particularly when compared to early spam filtering literature. Practically, filters trained with large corpora have a high potential impact on server-based spam filters, which is an important technique for efficient defense against spam (as reported by MAAWG [107]). At the same time, we recognize the need for future evaluation studies as even larger corpora become publicly available. With more than 460 billion messages processed by MAAWG in the first quarter of 2006, it is clear that we need larger and more up-to-date corpora for evaluating the effectiveness of evolving email spam filtering techniques. In addition, more

sophisticated statistical methods may be applied in the comparison of evaluation results on the slowly increasing number of published corpora for spam research.

CHAPTER IV

LARGE-SCALE EVALUATION OF EMAIL SPAM CLASSIFIERS

Learning filters [6, 9, 26, 46, 68, 70, 117, 127, 151, 156] have become a widely accepted technique for separating spam messages from an individual user's incoming email stream. Significant research and development efforts have produced software tools (e.g., SpamProbe, SpamBayes, etc.), which can be used by individuals to effectively filter spam. In addition to empirical use and anecdotal evidence, small-scale (on the order of a few thousand messages) evaluations of personalized learning filters have corroborated their effectiveness for individual use.

In this chapter, we present the first experimental evaluation of learning filter effectiveness using large corpora (about half a million known spam messages and a similar number of legitimate messages from public archives). Intuitively, the use of large corpora represents a "collaborative" approach to spam filtering. In contrast to filters trained by individuals with their own email, filters trained with large corpora are exposed to meaningful information from a variety of sources: legitimate messages from many individuals and spam messages from many sources to many destinations. In this chapter, we study the strengths and weaknesses of these collaborative filters.

Our first contribution in this chapter is a large-scale evaluation of learning filters that demonstrates their effectiveness with respect to known spam. In this evaluation, we observe that all of our filters achieve an accuracy consistently higher than 98%. However, we then show that this level of effectiveness is vulnerable to attacks using *camouflaged messages*, which combine both spam and legitimate content. Thus, our second contribution is a large-scale evaluation of the effectiveness of the camouflaged attack against our filters. This evaluation shows that the filters' accuracy degrades consistently to between 50% and 75%. In response to this attack, we designed and evaluated several strategies for increasing the resilience of the filters. Our third contribution is showing that a particular strategy works:

we can successfully identify camouflaged messages if the filters are explicitly trained to treat them all as spam. Using this strategy, the filters' accuracy climbs back to between 86% and 96%.

Our large-scale evaluations are particularly useful for companies and Internet service providers considering the adoption of learning filters in their email gateways. Our results support this "upstream" migration of spam filtering as a successful way to reduce bandwidth and storage costs associated with spam. Similar to the attacks on keyword-based filters, our evaluation of camouflaged attacks shows that spammers can easily lower the effectiveness of learning filters. However, unlike the attacks on keyword-based filters, we actually provide a strategy that trains the learning filters properly so that they can resist attacks.

The rest of the chapter is organized as follows. Section 4.1 summarizes our experimental setup and corpora processing. Section 4.2 illustrates the excellent performance of current spam filters with a normal training set and workload. Section 4.3 describes the degraded performance of our filters when they are under attack by camouflaged messages and provides a successful solution. Section 4.4 shows that our solution does not harm the excellent performance shown in Section 4.2. Section 4.5 discusses related work, and Section 4.6 summarizes our results.

4.1 Experimental Setup

4.1.1 Spam Filters

In our experiments, we used four different spam filters: Naïve Bayes [127], Support Vector Machine (SVM) [46, 147], LogitBoost [26, 61], and a Bayesian filter based on Paul Graham's white paper [68]. This collection of filters represents the state of the art in machine learning (SVM and LogitBoost) as well as the most popular and widely deployed spam filtering solutions (Naïve Bayes and Paul Graham-based). We chose the SpamProbe 1.1x5 software package [25] for our Paul Graham-based filter because it is highly configurable and very efficient. Additionally, an independent comparison of Paul Graham-based filters [37] found SpamProbe to be one of the most accurate filters of its kind. For our Naïve Bayes, SVM, and LogitBoost implementations, we chose the Weka software package [152]. Weka is an open

source collection of machine learning algorithms that has become a standard implementation tool in the machine learning community.

We used two Bayesian filters because each uses a different event model. SpamProbe is based on the multinomial model, and Weka's Naïve Bayes implementation uses the multi-variate Bernoulli model. Previous research [106, 134] has shown that the multinomial model typically outperforms the multi-variate Bernoulli model, and our results show that this phenomenon also holds with large corpora and in the presence of an attack. We used a linear kernel for our SVM filter because previous research [92] has shown that simple, linear SVMs usually perform as well as non-linear ones. We used 100 iterations for our LogitBoost filter because previous research [61] showed that using more than 100 iterations results in a miniscule improvement.

4.1.2 Corpora

Unlike early evaluation experiments involving spam filters [6, 9, 26, 46, 68, 70, 117, 127, 151, 156], which were user-specific or anecdotal, our experiments were not directly related to a specific individual's message set. Instead, based on our results in Chapter 3, we used very large, public corpora for our analysis.

Initially, we collected 750K spam messages from the publicly available spam corpora maintained by SpamArchive[1] and SpamAssasin[2]. After the spam messages were collected, 800K legitimate messages were collected from a number of sources. First, 4K legitimate messages were collected from the legitimate corpus maintained by SpamAssasin.[2] Then, another 4K legitimate messages were taken from the authors' personal mailboxes. Finally, a newsgroup crawler was used to obtain 792K messages from various publicly available newsgroups. Since a number of these newsgroups were not moderated, we used SpamProbe, trained with 8K randomly selected spam messages and the 8K legitimate email messages previously mentioned, to classify the newsgroup messages (refer to Section 4.2 for a validation of this method for filtering and labeling messages). The newsgroup messages that were labeled as spam by the filter were discarded, and afterwards, our collection of legitimate

[1]SpamArchive's spam corpora can be found at ftp://spamarchive.org/pub/archives/.
[2]SpamAssasin's spam and legitimate corpora can be found at http://spamassassin.org/publiccorpus/.

messages contained 700K messages.

In addition to the legitimate email corpus we collected, we also utilized the Enron corpus[3] [95]. However, in our initial experiments with this corpus, we discovered a number of messages that appeared to be spam. To alleviate this problem, we used SpamProbe, trained with 5K randomly selected spam messages and 5K randomly selected legitimate messages, to classify the Enron messages. The messages that SpamProbe labeled as spam were discarded, and then, a random sample of the remaining messages were manually inspected to ensure their legitimacy. After this process was complete, we were left with 475K Enron messages. Since the Enron corpus has become the standard legitimate email corpus amongst researchers, the results we present in this chapter were obtained using our cleansed version of this corpus.

Previous research [69, 156] claimed that significant performance improvements can be achieved by incorporating email headers in the filtering process. Our own experiments have also verified these results. However, we discovered that most of these performance improvements are caused by unfair biases in the corpora. An example of such a bias is the "Message-ID" header in the Enron corpus. Almost all of the messages in this corpus contain a "Message-ID" header, which contains "JavaMail.evans@thyme" as a substring. Consequently, if this header is included in the filtering process, it will unfairly bias each filter's evaluation of messages taken from our Enron corpus. Another example of a bias introduced by the corpora is the "Received" header in our SpamArchive corpus. More than 85% of the messages in this corpus contain at least one "Received" header, and most of those messages contain multiple "Received" headers. However, "Received" headers are not found in the messages in the Enron corpus. Consequently, if this header is included in the filtering process, it will unfairly bias each filter's evaluation of the messages taken from our SpamArchive corpus. Due to these and other biases introduced by the corpora, our experiments in this chapter used messages with all but two of their headers removed: "Subject" and "Content-Type."

[3]The Enron corpus can be found at http://www-2.cs.cmu.edu/ enron/.

A text message cannot be interpreted directly by some of our classifier building algorithms. Thus, an indexing procedure that maps a text message into a compact representation of its content needed to be uniformly applied to all of the messages in our corpora. All of our messages were tokenized using SpamProbe [25]. This tokenization resulted in each message being represented as a "set of words" (i.e., a tokenized message was represented as the set of tokens occurring in it). In previous research [10, 48], it was discovered that representations more sophisticated than this do not yield significantly better results.

4.2 Baseline Evaluation of Spam Filter Effectiveness

4.2.1 Cost Sensitive Performance Measures

To accurately evaluate our spam filters' performance, we need to incorporate the fact that classifying a legitimate message as spam (i.e., a false positive) is more costly to users than classifying a spam message as legitimate (i.e., a false negative). We are able to model this cost disparity using a performance metric called Weighted Accuracy (*WAcc*) [6]. *WAcc* assigns a weight of λ to legitimate messages, treating each legitimate message as if it were λ messages. Thus, if a legitimate message is (un)successfully classified, it counts as if λ legitimate messages are (un)successfully classified. The equation for *WAcc* is the following:

$$WAcc = \frac{\lambda \cdot n_{L \to L} + n_{S \to S}}{\lambda \cdot N_L + N_S}.$$

In the above equation, $n_{L \to L}$ is the number of correctly classified legitimate messages, $n_{S \to S}$ is the number of correctly classified spam messages, N_L is the total number of legitimate messages, and N_S is the total number of spam messages. In this chapter, most of the figures will demonstrate our filters' performance by showing their *WAcc* values with $\lambda = 9$.

4.2.2 Evaluation of the Training Set Size

To thoroughly investigate the effect of the training set size on our filters, we performed an experiment with five different training set sizes: 500, 1K, 5K, 10K, and 50K. First, we trained the filters with each training set, and then, we used the filters to classify a workload that consisted of 10K messages. Half of the messages used in the training sets and workload were randomly selected from our SpamArchive corpus, and the other half were randomly

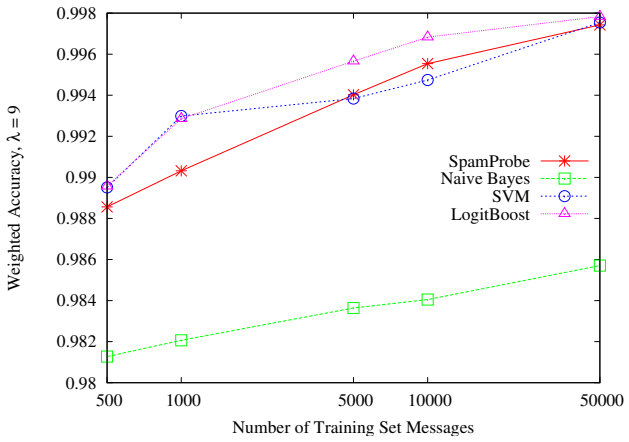


Figure 11: Average *WAcc* results ($\lambda = 9$) for the baseline evaluation using a 10K message workload and 640 retained features.

selected from our Enron corpus. Additionally, for these experiments, we retained 320 spam features and 320 legitimate features using the Information Gain feature selection algorithm.

Figure 11 illustrates the filters' average *WAcc* results for $\lambda = 9$ after 10 iterations of this experiment. SpamProbe, SVM and LogitBoost all exhibit similar performance, but Naïve Bayes performs slightly worse than the others. As the number of messages in the training set increases, the filters' performance also increases. However, for training sizes larger than 10K, we found that the filters' performance improves only modestly, failing to justify the additional training time necessary to use those larger training sizes. Consequently, for the remainder of this chapter, unless stated otherwise, we used 10K as the training set size for our experiments. We now turn our attention to the workload size.

4.2.3 Evaluation of the Workload Size

To analyze the effect of the workload size on our filters, we performed an experiment with five different workload sizes: 500, 1K, 5K, 10K, and 50K. First, we trained our filters with a training set of 10K messages. Then, we used the filters to classify the various workloads. Half of the messages in the training set and workloads were spam, and the other half were legitimate. Additionally, we retained 640 features (320 legitimate and 320 spam).

Figure 12 shows the filters' average *WAcc* results for $\lambda = 9$ after 10 iterations of this experiment. For smaller workload sizes (i.e., 500, 1K, and 5K), the filters demonstrate

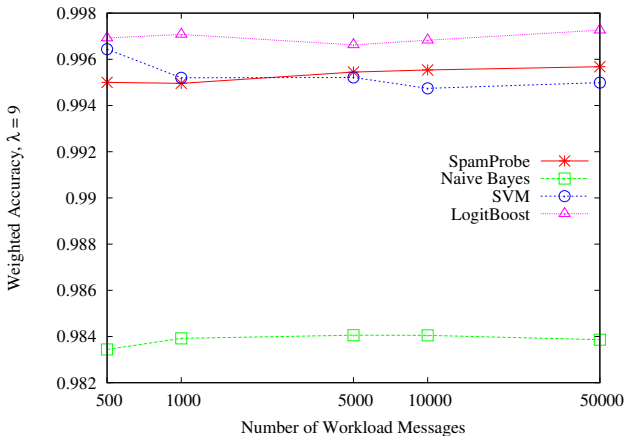


Figure 12: Average *WAcc* results ($\lambda = 9$) for the baseline evaluation using a 10K message training set and 640 retained features.

slightly fluctuating results; however, when the workload size is greater than 5K, the filters' performance is extremely stable. Additionally, the similarities between the results for 10K messages and 50K messages indicate that our corpora are internally consistent. Based on these results, we chose to use a workload size of 10K messages for the remaining experiments in this chapter. Now that we have experimentally determined an appropriate training set and workload size, we will investigate various feature sizes.

4.2.4 Evaluation of Feature Size

In the previous two experiments, our filters used 640 retained features. Our next experiment explores the effect of the number of retained features on the filters by using seven different feature sizes: 10, 20, 40, 80, 160, 320, and 640. In this experiment, we trained the filters with a training set of 5K spam messages and 5K legitimate messages. Then, we used the filters to classify a workload of 5K spam messages and 5K legitimate messages. The only variable was the total number of retained features.

Figure 13 displays the filters' average *WAcc* results for $\lambda = 9$ after 10 iterations of this experiment. All of the filters are quite successful when classifying the spam and legitimate messages in the workload. SpamProbe and Naïve Bayes both demonstrate their best performance with lower feature sizes, but SpamProbe's performance is consistently better. SVM and LogitBoost exhibit very similar behavior, and they both perform significantly better

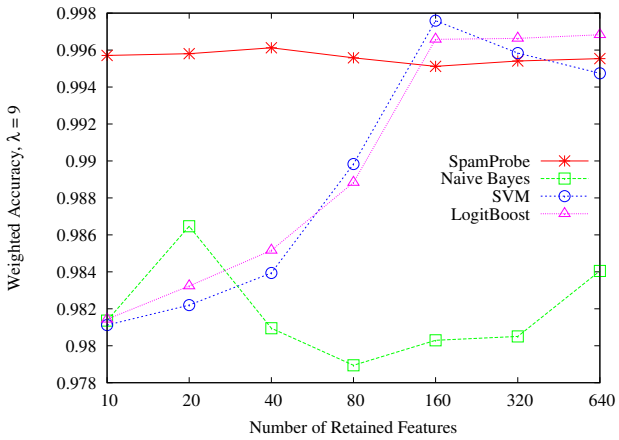


Figure 13: Average *WAcc* results ($\lambda = 9$) for a 10K message baseline training set and a 10K message baseline workload.

with larger feature sizes (i.e., more than 80 features).

These results concretely demonstrate the filters' ability to correctly classify the spam and legitimate messages in our large email corpora. However, in the next section, we present an attack, which significantly degrades our filters' classification performance.

4.3 Evaluation of Spam Filter Effectiveness Against Camouflaged Messages

4.3.1 Camouflaged Messages

The baseline results presented in Section 4.2 clearly illustrate the effectiveness of learning filters when distinguishing between spam and legitimate messages. However, we have recently observed a new class of messages that our filters are unable to correctly classify. These new messages, which we will refer to as camouflaged messages, contain spam content as well as legitimate content, and as a result, they are able to confuse our filters. The following sections describe experiments that quantitatively evaluate the influence of camouflaged messages on our learning filters. In these experiments, we vary the amount of camouflage (i.e., the relative proportions of the spam and legitimate content) used in the messages, and we also vary how the camouflaged messages are used in the training set and workload.

The camouflaged messages used in these experiments were generated using messages from the original training set of 10K messages and original workload of 10K messages used by the baseline experiment in Section 4.2.4. Each camouflaged message was created by

combining parts of a spam message with parts of a legitimate message. For each pair of spam and legitimate messages $\{s,\ l\}$, two camouflaged messages $\{c_1,\ c_2\}$ were created. c_1 was created by combining a larger portion of spam content with a smaller portion of legitimate content, and c_2 was created similarly by combining a larger portion of legitimate content with a smaller portion of spam content. In the experiments for this chapter, c_1 (c_2) contained twice as much spam (legitimate) content as legitimate (spam) content.

Before continuing to our experimental evaluation, an important observation must be made about the experiments involving camouflaged messages. In the baseline experiments, the messages were taken from known collections of spam and legitimate messages, and as a result, the training and evaluation of the filters was straightforward. However, it is unclear how we should determine the "spamicity" or "legitimacy" of camouflaged messages because they contain both spam and legitimate content. This uncertainty becomes even more pronounced as the amount of legitimate content found in these messages increases from 0% to 100%. At what point do we "draw the line" to distinguish between a spam message and a legitimate message? To address this question, we present a threshold t, which is used to quantify the amount of legitimate content a camouflaged message must contain in order to be identified as a legitimate message. A camouflaged message is identified as legitimate if and only if the proportion of its legitimate content is greater than or equal to t. Using this heuristic, we find that three distinct scenarios emerge. When $t = 0\%$, all camouflaged messages are treated as legitimate; when $0\% < t < 100\%$, the camouflaged messages are identified based on the proportion of their spam and legitimate content, and when $t = 100\%$, all of the camouflaged messages are treated as spam.

We exhaustively explored the design space for all three of these scenarios. By treating all camouflaged messages as spam (i.e., when $t = 100\%$) and carefully selecting the training set, we show in Section 4.3.2.2 that the filters can accurately identify camouflaged messages. However, similar results are not achieved in the other two scenarios.

4.3.2 Identifying All Camouflaged Messages as Spam

In this section, we evaluate our filters' ability to identify all camouflaged messages as spam using a series of experiments.

4.3.2.1 Original Training with Camouflaged Workload

In our first experiment, using a training set of only original messages, we analyzed the filters' ability to correctly classify a workload consisting entirely of camouflaged messages. Each filter was trained with the original training set of 10K messages used in the baseline experiment. Then, the filters were used to classify a workload consisting of 10K camouflaged messages, which were created using the original workload of 10K messages used in the baseline experiment in Section 4.2.4.

When we identify all of the camouflaged messages in the workload as spam, *WAcc* simplifies to spam recall because $n_{L\to L}$ and N_L are zero[4]. As a result, we use spam recall as the evaluation metric for these camouflaged experiments.

Figure 14 shows the filters' average spam recall results after 10 iterations of this experiment. This figure clearly shows that our original filters are unable to successfully classify the camouflaged messages in the workload as spam. SpamProbe consistently outperforms the other filters, but its best average spam recall value is only 76.57%. The Naïve Bayes filter performs the worst, consistently classifying only around 50% of the messages as spam. As in previous experiments, SVM and LogitBoost mimic each other's performance, but their highest average spam recall values are only 63.38% and 64.19%, respectively.

Obviously, when our filters only use original messages in their training set, they are incapable of correctly classifying the camouflaged messages in the workload as spam. Thus, to improve this situation, we performed additional experiments in which we varied the filters' training sets.

[4] *WAcc* simplifies to spam recall:

$$WAcc = \frac{\lambda \cdot n_{L\to L} + n_{S\to S}}{\lambda \cdot N_L + N_S}$$

$$\frac{\lambda \cdot (0) + n_{S\to S}}{\lambda \cdot (0) + N_S} = \frac{n_{S\to S}}{N_S} = spam\ recall$$

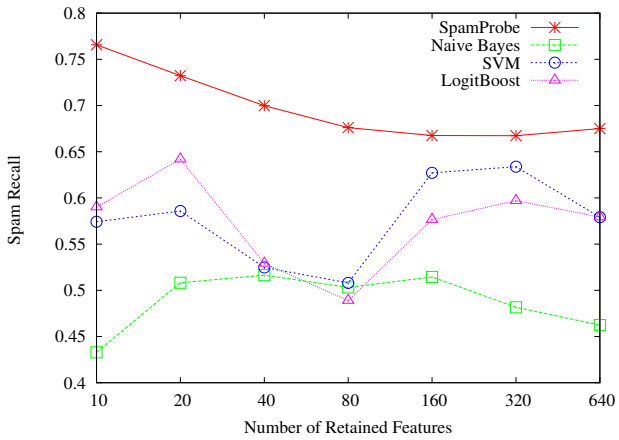


Figure 14: Average spam recall results for a 10K message baseline training set and a 10K message camouflaged workload.

4.3.2.2 Camouflaged Training with Camouflaged Workload

Our next experiments attempt to correctly classify camouflaged messages by introducing camouflaged messages into the filters' training sets. By training the filters with camouflaged messages, we expected to increase the filters' ability to successfully recognize camouflaged messages in the workload. To evaluate this expectation, we conducted two experiments.

The first experiment used a training set of 10K camouflaged messages, which were created using the original training set of 10K messages used in the baseline experiment in Section 4.2.4. Initially, the filters were trained to recognize all of the camouflaged messages as spam. Then, the filters were used to classify the same workload of 10K camouflaged messages used in the previous experiment in Section 4.3.2.1.

We omit this experiment's results because they are virtually identical to the results presented in the next experiment. Also, the filters created in this experiment are unable to correctly identify legitimate messages because they have only been trained to recognize camouflaged messages as spam. We save the discussion of this second point for Section 4.4.

The second experiment used the same camouflaged training set as the previous experiment, but it also used the original training set of 10K messages used in the baseline experiment. Thus, the filters were trained with 10K original messages as well as 10K camouflaged messages. Then, the trained filters were used to classify the same camouflaged

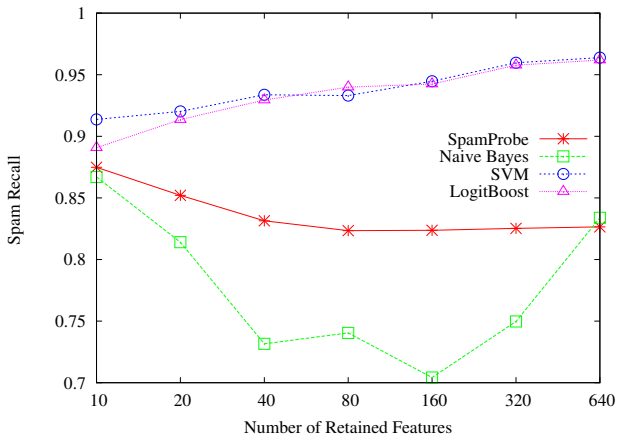


Figure 15: Average spam recall results for a 10K message baseline training set, a 10K message camouflaged training set, and a 10K message camouflaged workload.

workload used in the previous two experiments.

Figure 15 displays the filters' average spam recall values after 10 iterations of this experiment. By comparing these results to Figure 14, we see that all of the filters drastically improved their average spam recall values in this experiment. SpamProbe exhibited the smallest performance increase, but its best average spam recall value is still 87.49%. The Naïve Bayes filter improved the most, increasing its best average spam recall value from 51.64% to 86.70%. SVM and LogitBoost both perform extremely well, correctly identifying more than 96% (96.38% and 96.19%, respectively) of the camouflaged messages in the workload as spam.

These results show that if we train our filters to treat all camouflaged messages as spam, they will successfully classify the camouflaged messages in the workload as such. Thus, we have discovered a solution for successfully identifying camouflaged messages.

4.4 Baseline Evaluation of Spam Filter Effectiveness Revisited

In Section 4.3, we evaluated techniques for classifying camouflaged messages in the filters' workloads. In this section, we take that evaluation a step further in order to determine the effect our proposed solutions have on each filter's baseline performance with the original workload. If one of the proposed solutions sacrifices the filters' ability to classify messages in the original workload, its overall worth is significantly discounted.

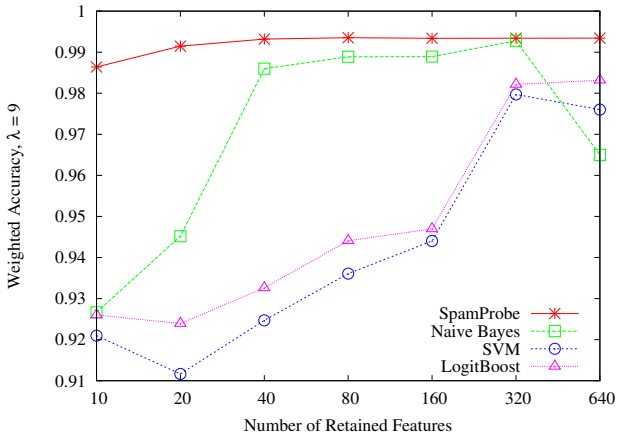


Figure 16: Average *WAcc* results ($\lambda = 9$) for a 10K message baseline training set, a 10K message camouflaged training set, and a 10K message baseline workload.

The series of experiments we used to analyze each proposed solution's effect on the baseline performance is very similar to the series of experiments performed in Section 4.3.2.2. The training set used in each of the experiments in this series is the same as the training set in the corresponding experiment in the previous series. However, instead of using the camouflaged workload of 10K messages used in the previous series, this series' experiments used the original workload of 10K messages used in the baseline experiment.

The first experiment in this series used the filters that were only trained to treat all camouflaged messages as spam. We omit the figure for this experiment's results because the filters created in this experiment perform so poorly with the baseline workload. As explained in Section 4.3.2.2, these filters are unable to correctly classify legitimate messages because they have only been trained to identify spam. Since this proposed solution dramatically reduces the filters' baseline performance, it must be rejected.

The results obtained in the first experiment highlight an important point about our proposed solutions. To be a truly effective solution, the training method must create filters, which successfully classify camouflaged messages while also maintaining the filters' baseline performance. With that in mind, we now describe our final experiment.

The last experiment used the filters that were trained with the original training set and trained to treat all camouflaged messages as spam. Figure 16 shows the average *WAcc* results for $\lambda = 9$ after 10 iterations of this experiment. By comparing Figures 16 and 13, we

discover that the average *WAcc* values for the filters in this experiment are very similar to the corresponding values in the baseline experiment. SpamProbe's performance experienced almost no change from the baseline experiment. SVM and LogitBoost experienced a modest performance decrease, but the decline in performance was only a couple percentage points.

These results illustrate the filters' ability to correctly classify the original workload when they are trained with the original training set and trained to treat all camouflaged messages as spam. Additionally, since we concluded in Section 4.3.2.2 that these filters could correctly classify camouflaged messages as spam, we have experimentally proven that this proposed solution is truly effective at accurately classifying both camouflaged and original messages.

4.5 Related Work

The use of machine learning algorithms in spam filters is a well established idea. Previous research [6, 9, 117, 127] has shown that the Naïve Bayes algorithm can be used to build very effective personalized spam filters. Similar results have also been obtained for Support Vector Machines [9, 46, 156] and Boosting algorithms [9, 26, 156]. However, our work is unique in that we study the effectiveness of spam filtering on a very large scale. We used corpora that are two orders of magnitude larger than those used in previous evaluations, and as a result, we have shown that machine learning techniques can be used to successfully filter spam at the gateway.

We are not the first to suggest that learning filters are potentially vulnerable to attack. Graham-Cumming gave a talk at the 2004 MIT Spam Conference [70], in which he described an attack against an individual user's spam filter. This attack inserted random words into spam messages. A previous paper [151] attempted to expand this attack by adding commonly occurring English words instead of random words. Our research differs from this previous work in a number of ways. First, our described attack (i.e., camouflaged messages) uses portions of real legitimate email, and as a result, it creates more confusion than the previously mentioned attacks. Second, our evaluation incorporated a larger set of learning filters. Finally, we focused our attention on gateway filtering, not user-specific filtering.

Attacking a spam filter can be thought of as an example of how machine learning

algorithms can be defeated when an adversary has control of the workload. Thus, this chapter is similar in spirit to [42], in which the authors used game theory to model attacks on spam filters. In that paper, the emphasis was placed on the game theoretic modeling; however, we showed that in the domain of spam filtering, a spammer can successfully attack current filters without using sophisticated game theoretic methods.

4.6 Summary

In this chapter, we used large corpora (half a million known spam and about the same number of legitimate messages from public collections) to evaluate learning filter effectiveness in spam filtering. The use of such large corpora represents a collaborative approach to spam filtering because it combines many sources in the training of filters. First, we evaluated a production-use filtering tool (SpamProbe) and three classic machine learning algorithms (Naïve Bayes, Support Vector Machine (SVM), and LogitBoost). We found these filters to be highly effective in identifying spam and legitimate email, with an accuracy above 98%. Second, we described a simple method of attacking these learning filters with camouflaged messages, which significantly lowered the effectiveness of all the learning filters in our large-scale experiments. Third, by exploring the design space of training strategies, we found that by treating all camouflaged messages as spam during the training phase, we can restore the filters' effectiveness to within a couple percentage points of their baseline accuracy.

CHAPTER V

EVOLUTIONARY CHARACTERISTICS OF EMAIL SPAM

The goal of spam producers is to create spam messages that reach their intended receivers (called victims or simply users). In response to the increasing amount of spam, many victims have adopted statistical learning filters [9, 68, 127, 156] with the goal of finding and "killing" spam before it reaches their mailboxes. These frontally opposing goals have been modeled as an arms race [42, 104], with the evolution of spam construction techniques and the increasing sophistication of spam filtering techniques. Our study on spam evolution is inspired by an analogy between the spam arms race and the biological arms race, where new drugs (e.g., antibiotics) are created to kill existing bacteria as well as the subsequent evolution of new bacteria variants that are capable of resisting these new antibiotics.

In this chapter, we describe a population evolution study of spam construction techniques based on their "genetic markers" in spam messages. Specifically, we adopt the detailed analysis of spam message content and structure developed and maintained by the SpamAssassin Project [140]. In SpamAssassin 3.1.0, 495 *spamicity tests* are used to characterize spam. These tests reflect specific spam construction techniques that are used by spam producers. Typically, these spam construction techniques are syntactically correct features that are rarely used in legitimate email but frequently abused by spam producers in the construction of many spam messages. Like genetic markers, the spamicity tests help characterize spam through a detailed analysis of message content and structure. However, unlike genetic markers that deterministically characterize a strain of bacteria, the spamicity tests are statistical in nature, only indicating a probability of whether the message is spam. We observe at the outset that we are not evaluating the effectiveness of SpamAssassin's approach (as a spam filtering technique). We simply use SpamAssassin's tests as a type of "genome mapping" in our study of spam evolution. Concretely, we will look at the prevalence of spam messages that employ specific spam construction techniques (i.e., they

test positive for specific spamicity tests) and analyze the changes in their popularity over a three-year period. This is analogous to a population study of bacteria strains using specific genetic markers. As a result, we sometimes refer to spam messages that test positive for a spamicity test as a "strain of spam."

In this chapter, we study two trends in the analysis of spam construction techniques. The first trend of interest is *extinction*, indicated by the population of a strain of spam declining to zero or near zero during the study period. We will attempt to find a causal explanation for the spamicity tests that show extinction of spam messages employing those spam construction techniques. The second trend of interest is *co-existence*, indicated by a sustained population of a strain of spam, particularly through the end of the study period. Co-existence shows the survival of some spam construction techniques, even though the presence of spamicity tests shows a clear ability to identify those strains of spam messages. We found that explaining co-existence was usually quite speculative at this stage of study.

In our trend analysis of spam construction techniques, we classify the spamicity tests (and our explanations) into three groups of significant factors in our study of spam evolution. These groups are: (1) significant environmental changes, (2) individual filtering techniques, and (3) collaborative filtering techniques. For the cases of extinction, our hypothesis is that the identification of that spam construction technique (i.e., the definition of that spamicity test) was the cause of extinction. Conversely, the long term persistence of a strain of spam would indicate that the corresponding spamicity test did not cause the extinction of the strain, even though that spam construction technique is clearly identifiable. The co-existence of a persistently surviving spam construction technique and its spamicity test in spam filters indicates an equilibrium similar to the co-existence of a pray and its predator. The co-existence does not necessarily mean the predator is ineffective in killing some of the pray. It simply indicates some concrete limitations in the predator's killing capability that allows the pray to continue to survive and perhaps even thrive. This study does not evaluate quantitatively the "amount of killing" for each spamicity test, which is a subject of future research.

The main contribution of the chapter is the large-scale experimental evaluation of the

prevalence of representative spam construction techniques over a three-year period. Concretely, we study the evolution of more than 1.4 million spam messages that were collected from January 2003 through December 2005. Through this study, we have found convincing evidence that some factors have been effective in causing the extinction of specific strains of spam. As an example of significant environmental changes in the extinction category, the removal of USERPASS support by Internet Explorer and Mozilla in 2004 seems to have effectively eliminated that feature from spam messages. Prior to this environmental change, spammers (and phishers, in particular) were exploiting a syntactic feature of URLs (i.e., the ability to include arbitrary text in the <user>:<password> field of a URL) that allowed them to deceive users.

Perhaps more intriguingly, we failed to find conclusive evidence of extinction where some was expected. For example, URL block lists are considered an effective method for identifying spam-related URLs. When they are adopted by collaborative filtering, they are a powerful technique to identify spam [31, 119]. However, Figure 26 shows that despite the deployment of block lists by many sites, a significant percentage of spam messages persist in containing URLs that are listed on the block lists. Therefore, we include the URL block lists in the co-existence category. We note the coarse granularity of our study, which is only concerned with the extinction or co-existence of a particular spam strain. Consequently, URL block lists could be effective in distinguishing a number of spam messages, but they have not been as strong a deterrent as the removal of USERPASS support by browsers.

Our study based on spamicity tests has goals and methods that are qualitatively different from most previous and current reports on spam evolution [22, 23, 50, 85, 138], which focus primarily on the evolution of spam content (e.g., the emergence and popularity of certain topics such as drugs and stocks). By focusing on spamicity tests, our goal is to learn more about what allows some spam messages to pass through the filters to reach their victims and what prevents others. This is in contrast to topical analysis, which is primarily a reflection of the expected economic gains of spam producers.

The rest of the chapter is organized as follows. Section 10.3 describes our spam corpus and experimental setup. Section 5.2 shows illustrative examples of spamicity tests that

became extinct over time. Section 5.3 summarizes examples of spamicity tests that show unexpected resilience to filtering techniques. Section 5.4 summarizes related work, and Section 5.5 summarizes our findings.

5.1 Experimental Evaluation Method

5.1.1 Spam Corpus Collection and Preparation

Since January 2003, we have been collecting spam messages systematically. For each period (e.g., monthly), we copy the new spam messages from the SpamArchive spam corpora[1]. As of January 2006, our accumulated spam corpus contained more than 1.4 million spam messages. SpamArchive's spam messages are stored in two collections: submit and submitautomated. The messages in the submit collection were submitted individually by users, and the messages in the submitautomated collection were submitted by automated tools on behalf of their users. Each of these collections contains close to a thousand archives that are stored as gzipped mbox folders. The spam messages within these mbox folders contain most of their original headers; however, some information has been removed to protect the privacy of the users that submitted the messages. Specifically, the recipient of the message (the "To" header) has been replaced by "submit@spamarchive.org," and the IP address of the relay recipient in the first "Received" header (i.e., the relay used to deliver the message to the submitting user) has been omitted.

Since the SpamArchive spam corpora are updated daily, our system is fully automated to update concurrently with those corpora. Every day, the system performs a number of activities. First, it downloads the latest archives from SpamArchive's two spam collections (i.e., submit and submitautomated), and these archives are gunzipped to obtain the corresponding mbox folders. Next, to facilitate tracking the evolution of various spam characteristics over time, the spam messages in these mbox folders are sorted based on the month and year they were received by the users that submitted them to SpamArchive.

The email messages stored in an mbox folder typically have three fields that store date information: the "From " line that delimits each message in the mbox folder (not to be

[1]SpamArchive's spam corpora can be found at ftp://spamarchive.org/pub/archives/.

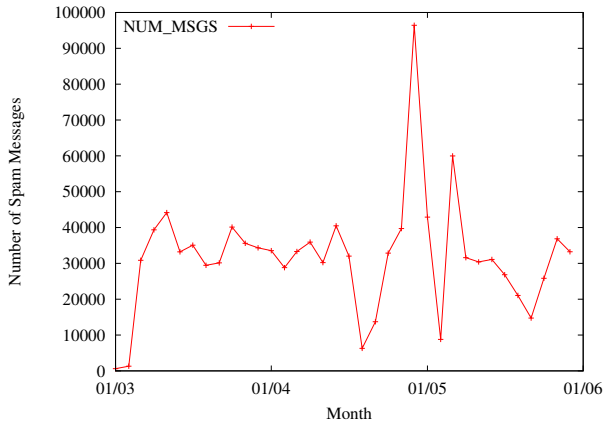


Figure 17: Month-by-month break-down of the number of spam messages in our spam corpus.

confused with the "From" email header), the "Date" email header, and the "Received" email header(s). To sort the spam messages, we used the date information found in their "Received" email header(s) because that information is the most reliable indication of when the messages were delivered to the submitting users. Specifically, we used the first "Received" header because it is attached to the message by the relay that is responsible for delivering the message to the submitting user (i.e., the most trustworthy relay between the spammer and the end user). We rejected the date information in the "From " line because it represents the date that the message was placed in the mbox folder by SpamArchive and not the date that the message was received by the submitting user. We rejected the "Date" email header date information because it can be spoofed easily by spammers. Figure 17 shows the distribution of spam messages that were received from January 2003 through December 2005, based on our sorting algorithm.

5.1.2 Testing Infrastructure

As previously mentioned, the actual characteristics of spam messages that are used in our evaluation have been adopted from SpamAssassin 3.1.0. SpamAssassin is an open source spam filter that identifies spam messages by combining various spam detection techniques. These techniques include textual analysis of a message's headers and body, querying DNS block lists, and querying collaborative filtering databases.

Each of SpamAssassin's spam detection techniques is composed of a variety of spamicity tests. For example, SpamAssassin's textual analysis component contains tests such as OBFUSCATING_COMMENT and INTERRUPTUS, which identify examples of HTML-based obfuscation techniques. All of these tests have a user-specified score associated with them. When SpamAssassin analyzes a given message, it runs all of the spamicity tests. When the message satisfies one of the tests (i.e., it tests positive), that test's score is added to an overall spamicity score. The message is classified as spam if its accumulated overall spamicity score is above a user-specified threshold. In our experiments, we are only interested in the results of each test on each message, and we ignore the overall score since we are not using SpamAssassin as a spam filter. Specifically, we ran the spamicity tests on the messages (grouped by month), and for each test, we counted the number of messages (population) that tested positive. To compensate for the variations of new spam messages each month (Figure 17), we normalize the population count, dividing it by the total number of messages in that month.

5.1.3 Overview of the Spam Evolution Study

In this section, we summarize the results of evaluating all 495 spamicity tests from SpamAssassin 3.1.0 on our three-year spam collection from SpamArchive. We have divided the spamicity test graphs into three groups: extinction, co-existence, and complex. The extinction group (discussed in Section 5.2) consists of graphs that show a downward trend, starting from a significant number of messages testing positive and ending with a relatively negligible number during the last three months. The co-existence group (discussed in Section 5.3) consists of graphs that show a persistently high number of messages testing positive at least for the last three months (regardless of the starting point). The complex graphs combine different trends or contain high variability, and their analysis is the subject of future research.

The extinction group includes 236 spamicity tests, and the co-existence group includes 64 spamicity tests. We studied the distribution of test popularity within each group. Table 15 contains the distribution of the number of tests (divided by group), according to the

Table 15: Distribution of average results.

Average Range	Number of Spamicity Tests		
	Extinction	Co-existence	Complex
[0.0 – 0.1)	230	42	195
[0.1 – 0.2)	6	11	0
[0.2 – 0.6)	0	11	0

Table 16: Distribution of maximum results.

Maximum Range	Number of Spamicity Tests		
	Extinction	Co-existence	Complex
[0.0 – 0.1)	201	26	180
[0.1 – 0.2)	22	12	14
[0.2 – 0.3)	8	8	1
[0.3 – 0.4)	4	4	0
[0.4 – 0.5)	1	5	0
[0.5 – 0.9)	0	9	0

average percentages calculated over the three-year period. The overwhelming majority of tests averaged less than 10% of the messages, with only 22 tests in the co-existence group averaging between 10% and 60%. Since the co-existence group consists of curves that remain flat (to be considered surviving), it is expected that this group contains the highest average numbers. However, the average does not represent the extinction group since it blurs the downward trend that characterizes the group. Table 16 contains the distribution of the number of tests according to the maximum percentages achieved during that three-year period. The table shows that the extinction group contains a significant number of tests (35) that started with more than 10% of messages containing that spam construction technique.

The remaining graphs (Figure 18 through Figure 27) shown in the chapter are population evolution graphs, with the x-axis representing time (from January 2003 through December 2005) and the y-axis representing the percentage of messages (in a given month) that tested positive for the various spamicity tests being shown in the graph.

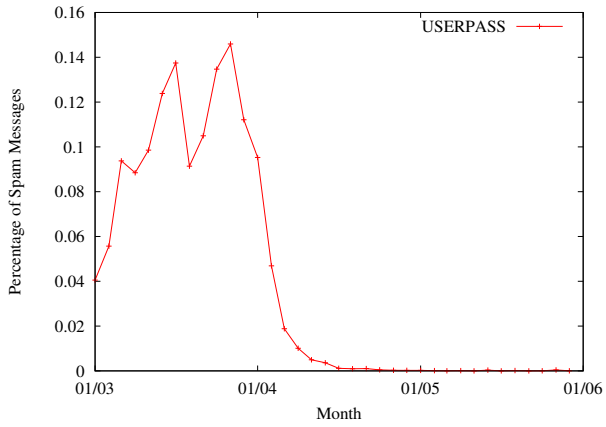


Figure 18: Evolution of the presence of Username:Password URLs.

5.2 Evidence of Extinction

Of the 236 graphs in the extinction group, we selected a few of the most interesting ones for discussion. In a sense, these are the "success stories" for spam filtering or other techniques that combat spam since spam producers are completely avoiding these markers.

5.2.1 Significant Environmental Changes

The evolution of the USERPASS spam signature is an example of the extinction category. According to RFC 1738 [19], URLs with the following format are syntactically valid:

<scheme>://<user>:<password>@<host>:<port>/<url-path>

However, spam producers (particularly, phishing message producers) began to exploit this URL format to deceive users, disguising their malicious sites by inserting a popular site in the <user>:<password> field. For example, a phisher might use http://www.ebay.com@badsite.com to trick users into believing they're accessing ebay.com, when they're actually accessing badsite.com. A more extensive discussion of this technique (and its many variations) is provided in [116]. The USERPASS spam signature tracks messages that contain URLs with the <user>:<password> format. As Figure 18 shows, more than 10% of the spam messages in almost every month from May 2003 through January 2004 contained at least one USERPASS URL. Then, rather abruptly, only 4.7% of the spam messages in February 2004 contained a USERPASS URL. In the following month, this percentage fell to 1.9%,

and by May 2004, almost none of the spam messages contained a USERPASS URL. Upon observing this trend, the most obvious question is, "What forced phishers to abandon this technique?"

On February 2, 2004, Microsoft issued Microsoft Security Bulletin MS04-004[2] along with a security update that removed support for USERPASS URLs from Internet Explorer. The Mozilla Project quickly followed suit by removing USERPASS support from Mozilla (as described in Mozilla's Bugzilla Bug 232567[3]). Both of these actions are environmental changes that made the USERPASS option useless to spam/phishing producers. As the figure shows, by mid-2004, spam producers eliminated all USERPASS URLs from their messages.

5.2.2 Individual Filtering

One of the earliest defenses against spam was keyword-based filters [35]. Unfortunately, spam producers defeated keyword filters by replacing keywords with randomized misspellings. In response, victims began to use statistical learning filters (e.g., Naïve Bayesian, Support Vector Machines – SVM, and LogitBoost) [9, 68, 127, 156] that are capable of learning and identifying a large number of unpredictable misspellings. These filters operate individually (i.e., they are trained by each user), and it appears that some spam strain extinctions are due to the effectiveness of these individual filters. An example is HTML-based obfuscation techniques.

We first discuss the evolution of four spam construction techniques involving HTML-based obfuscations:

- OBFUSCATING_**COMMENT**
- **INTERRUPTUS**
- HTML_FONT_**LOW_CONTRAST**
- HTML_**TINY_FONT**

[2] http://www.microsoft.com/technet/security/bulletin/MS04-004.mspx

[3] https://bugzilla.mozilla.org/show_bug.cgi?id=232567

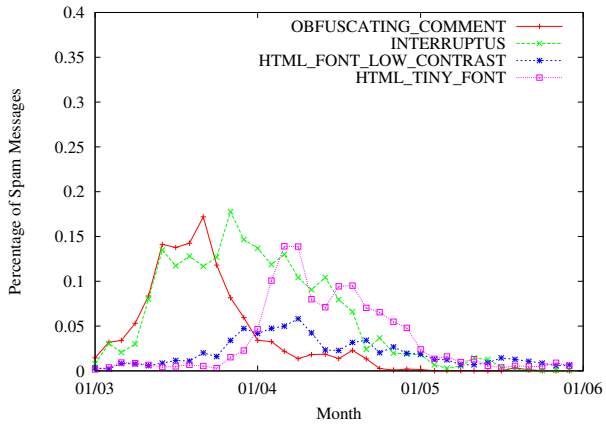


Figure 19: Evolution of specific HTML-based spam obfuscation techniques.

These techniques are used to disguise keywords that indicate spam (have high spamicity scores). Each one of the techniques can be used to invisibly divide spam keywords into randomized components. The result is the avoidance of keyword filters and low spamicity scores by statistical learning filters since the customary spam keywords are never present in their entirety. For example, **LOW_CONTRAST** and **TINY_FONT** were used to introduce virtually invisible fragments so the visual presentation becomes quite different from the underlying parsed text. Similarly, **COMMENT** and **INTERRUPTUS** are used to insert HTML tags in the middle of keywords, making the keywords unrecognizable by learning filters. For example, spam producers might obfuscate the word *Viagra* using Vi<xxx>ag<yyy>ra or V<!−−x−−>iagr<!−−y−−>a.

Figure 19 shows the evolution of these four spam construction techniques. Initially, the **COMMENT** and **INTERRUPTUS** techniques were the most popular. Then, as the popularity of the **COMMENT** technique steadily declined, spammers focused their attention on the INTERRUPTUS technique. When the INTERRUPTUS technique began to decline (after November 2003), the **LOW_CONTRAST** and **TINY_FONT** techniques were already rising in popularity. These phase differences suggest an arms race between spam producers and individual filters. As spam producers adopt an HTML-based obfuscation technique, spam filters (e.g., SpamAssassin and statistical learning filters) begin to associate the technique with high spamicity scores during the continuous retraining of the filter. As we explained in the previous chapter, the effectiveness of filter retraining forces

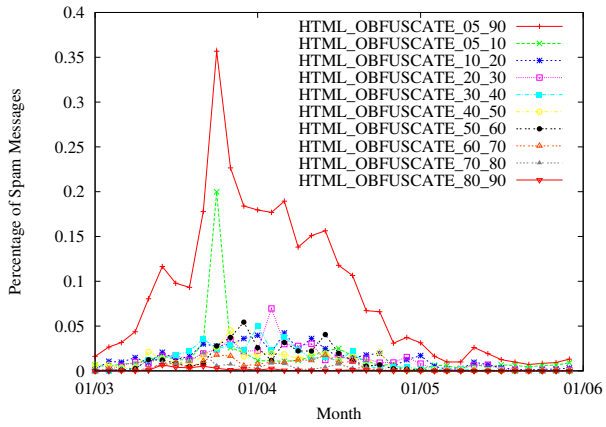


Figure 20: Evolution of all HTML-based spam obfuscation techniques.

spam producers to migrate to a new obfuscation technique. Figure 19 shows four specific examples of these arms race cycles.

Fortunately, the spamicity tests also give us a tool to analyze all HTML-based obfuscation techniques as a group. Figure 20 shows the population of spam messages that employ any HTML-based obfuscation techniques, classified by the percentage of HTML obfuscation content in each message. For example, the line marked as HTML_OBFUSCATE_05_10 represents the percentage of spam messages with a message body consisting of between 5% to 10% HTML obfuscation content. Similarly, the line for HTML_OBFUSCATE_05_90 represents almost all of the messages that contain HTML obfuscation content. In Figure 20, we can observe that the line for HTML_OBFUSCATE_05_90 gradually increases, indicating a possible learning curve. Then, after the peak in October 2003, spam producers began to slowly move away from HTML-based obfuscations, and by March 2005, the number of messages containing HTML obfuscation techniques became vanishingly small.

Figures 19 and 20 suggest that individual filters won a battle against spam producers in the HTML-based obfuscation arms race. Although new obfuscation techniques (e.g., camouflage attacks) continue to plague learning filters, the individual filters were able to successfully detect HTML-based obfuscation techniques. As a result, by the end of 2005, this filtering ability forced the spam producers to virtually abandon spam construction techniques using HTML-based obfuscation.

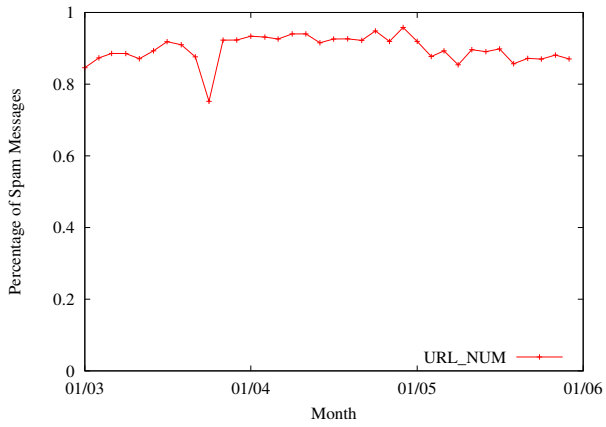


Figure 21: Evolution of the presence of URLs in spam messages.

5.2.3 Collaborative Filtering

Our efforts to find a clear example of extinction for spamicity tests in the collaborative filtering category failed to yield good results. Instead, we found some evidence of co-existence (Section 5.3.3), where clearly effective collaborative filtering techniques did not cause those spam construction techniques to become extinct. Whether collaborative filtering techniques have inherent limitations that prevent them from "killing off" some strains of spam is an interesting area of future research.

5.3 Survival and Co-Existence

In this section, we discuss examples of spam construction techniques that exhibit unexpected and persistent resiliency. These examples are interesting since they seem to work well for the spam producers, despite explicit identification tests and attempts to filter them out. This phenomenon is contrary to our expectations since we would normally expect the spamicity tests to be effective in filtering out these targeted spam messages. The resilient nature of these spam construction techniques makes them good candidates for further research since these strains of spam are surviving and perhaps even thriving.

5.3.1 Significant Environmental Changes

First, we discuss a spamicity test that, by definition, cannot be used to extinguish spam messages: the presence of URLs in an email message. In contrast to USERPASS, which

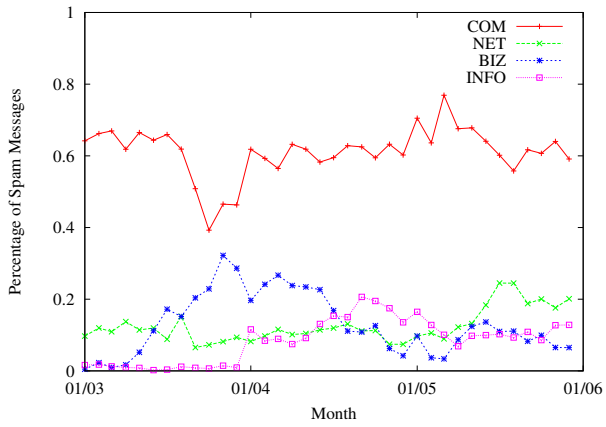


Figure 22: Evolution of the presence of URLs with specific TLDs.

was rarely used in general, URLs are often present in both spam and legitimate messages. Figure 21 shows that at least one URL appeared in between 85% and 95% of spam messages in every month except for October 2003 (when only 75% of the spam messages contained at least one URL).

Although a single data point could be the result of data collection problems or random statistical fluctuations, we have a conjecture for the dip shown in October 2003. On September 23, 2003, California Governor Gray Davis signed into law an anti-spam bill, Senate Bill No. 186, that made each email advertisement fineable up to $1 million [118]. This could explain the removal of URLs from some spam messages and the dip shown in October 2003. Unfortunately, Congress passed the CAN-SPAM Act at the end of October, which replaced the strict penalties detailed in the California anti-spam bill [90]. This could explain the "back to business as usual" mindset of spam producers and the restoration of URLs to their normal level.

A refinement of the URL-presence spamicity test consists of the tests for URLs with specific top level domains (TLDs). While COM and NET are commonly used TLDs, other TLDs such as BIZ and INFO have also been used frequently by email marketers. Figure 22 shows the evolution of spam messages containing URLs with four specific TLDs: COM, NET, BIZ, and INFO (the four most popular TLDs found during the three-year period). We observe a dip in the COM curve around October 2003 (lasting four months). This dip seems anti-correlated with a peak in BIZ around the same time. Whether this valley and

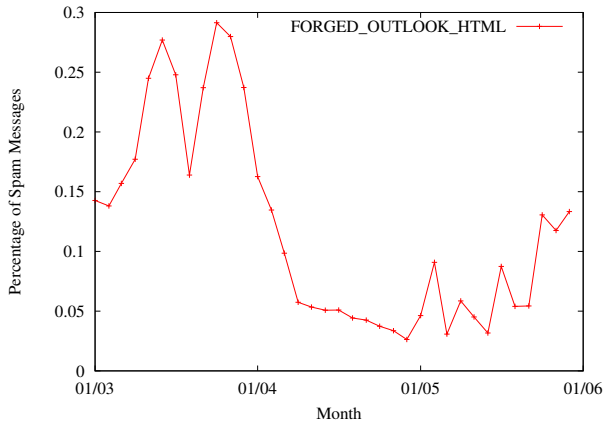


Figure 23: Evolution of messages that pretend to be sent using Outlook.

peak are correlated with the dip in Figure 21 is open for debate. Another observation from Figure 22 is that the spam producers' favorite TLD choice (other than COM) has changed from NET (January 2003 to June 2003) to BIZ (July 2003 to July 2004) and INFO (August 2004 to March 2005), with NET eventually returning to the top position (April 2005 to December 2005).

5.3.2 Individual Filtering

In this section, we analyze three cases of individual filtering that were unsuccessful, despite seemingly having the capability to extinguish spam messages of a particular strain. The first case concerns spam messages that pretend to be sent by Outlook (i.e., the messages contain a forged "X-Mailer" header with "Microsoft Outlook"). When the real Microsoft Outlook application sends an HTML email message, two versions of the message are sent: the HTML version and an automatically generated plain text version. Since the forged messages only contain HTML content, they could not have been sent by Outlook; thus, this spamicity test is a fairly reliable indicator of spam. Figure 23 shows the survival and co-existence of spam messages containing the forged Outlook header with this spamicity test. In 2003, the forged header was consistently identified in over 15% of the spam messages, but this value dropped to around 5% in 2004. Surprisingly, in 2005, the technique began to grow in popularity towards 15%, despite the spamicity test.

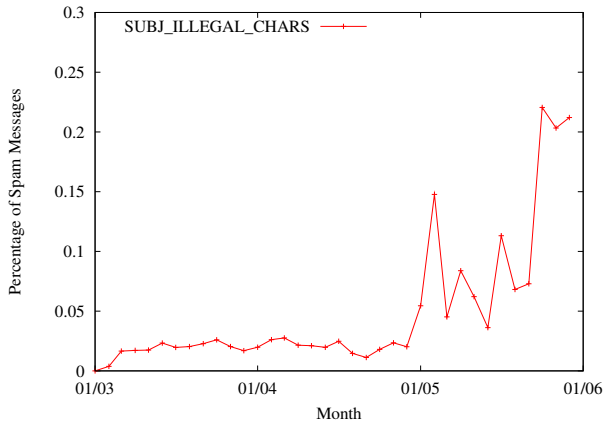


Figure 24: Evolution of messages that use at least 2 illegal characters in their "Subject" headers.

The second case consists of messages that use at least 2 illegal characters in their "Subject" headers. An illegal character is defined as a character that should be MIME encoded (as per RFC 2045 [58]) but is not. Figure 24 shows the evolution of the number of messages that employ this spam construction technique. This figure shows that despite the identification of this spamicity test and attempts to filter it out, the number of spam messages containing such illegal characters actually grew from about 2% before 2005 to about 20% at the end of 2005. The reasons for this thriving spam construction technique are a subject of future research.

The third case consists of messages that contain a specific pattern in their "Message-ID" headers. The pattern is defined by the following Perl regular expression:

```
<[0-9a-f]{4,}\$[0-9a-f]{4,}\$[0-9a-f]{4,}\@\S+>
```

This pattern is legitimately used by mail clients that place "Produced By Microsoft MimeOLE" in their "X-MIMEOLE" headers. Thus, if a message contains the pattern without this value in its "X-MIMEOLE" header, the "Message-ID" header is considered forged, and the message is considered spam. Figure 25 shows the evolution of the number of messages that have the above pattern in their "Message-ID" headers. This feature has gained and lost popularity over the years, with a low of 2% in early 2005, followed by a sudden growth during 2005, and another low of 2% at the end of 2005. Due to the ups and

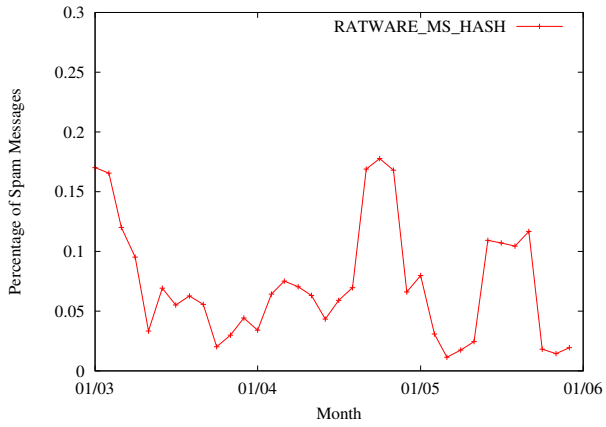


Figure 25: Evolution of messages that have a specific pattern in their "Message-ID" headers.

downs in the figure, despite the ease of executing this spamicity test, whether this strain of spam has become extinct is unclear. Depending on the interpretation of the curve during 2005, Figure 25 can be interpreted as extinct (if you only look at the last three months), co-existence (if you take the average for the year), or inconclusive and therefore in the complex category.

5.3.3 Collaborative Filtering

Another example of an obviously effective spamicity test concerns "URL block lists" that enumerate URLs that are known to be spam-related through reliable sources. These block lists are typically constructed and maintained by collaborative filtering (i.e., contributions by many trusted participating users). For a given block list, the spamicity test finds the spam messages that contain at least one URL that appears on that block list. Figure 26 shows the evolution of the number of messages that contain at least one URL that appears on one of three block lists: ws.surbl.org, jp.surbl.org, and ob.surbl.org.

As the figure shows, the percentage of spam messages that contained at least one URL on any of the three block lists remained below 20% up until March 2004. Then, from March 2004 through August 2004, the percentage of spam messages that contained at least one URL on ws.surbl.org grew from 21.4% to 71.9%. It took jp.surbl.org slightly longer to gain this level of popularity, but from March 2004 through October 2004, the percentage of spam messages that contained at least one URL on this block list grew from 17.5% to 78.6%. The

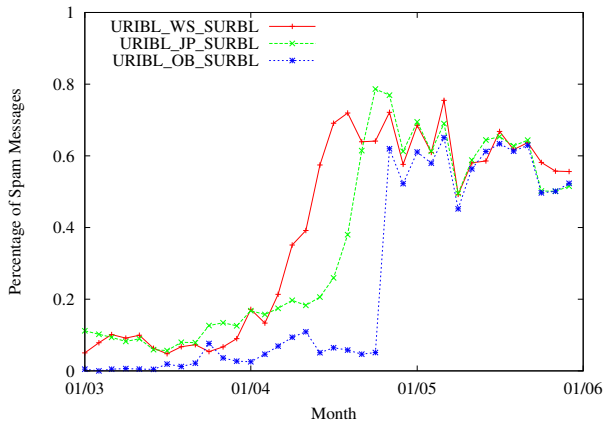


Figure 26: Evolution of URLs appearing on block lists.

ob.surbl.org block list was the slowest to obtain popularity, but from October 2004 through November 2004, the percentage of spam messages that contained at least one URL on this block list skyrocketed from 5.1% to 62.0%.

An alternative explanation for the population gains in Figure 26 is the improvement in collaborative filtering performance. Suppose that the effectiveness of collaborative filtering is directly related to the participation of effective collaborators. It is reasonable to assume that at the beginning of any collaborative effort, only a limited number of effective collaborators participate, with a limited coverage. In the case of block lists, this effect would translate to a partial coverage of known spam-related URLs. As more and more people contribute suspicious URLs, the block list becomes more comprehensive, and the spamicity test becomes more effective. This may explain the phase differences among the three lists, if we assume that ws.surbl.org achieved full effectiveness first, followed by jp.surbl.org and ob.surbl.org. Figure 26 shows that the block lists are clearly effective in identifying spam messages (finding more than 50% of the spam messages after November 2004) that contain spam-related URLs.

As we acknowledge the success of URL block lists, the continued and constant presence of spam-related URLs in spam messages also shows that the effectiveness of block lists is somehow limited. Unlike USERPASS and HTML-based obfuscation, which became extinct by a change in browser technology and effective individual filtering, the presence of spam-related URLs on block lists did not completely "kill" this strain of spam. The survival

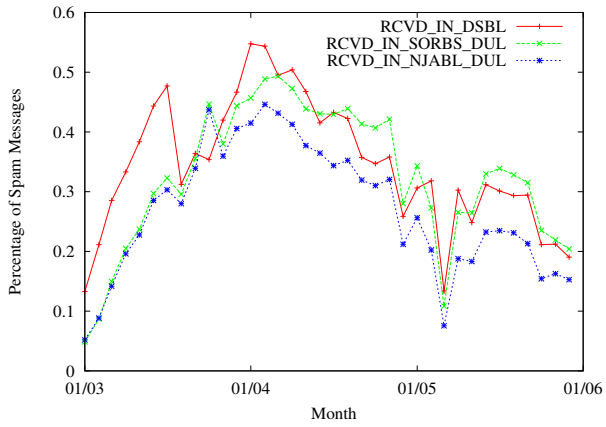


Figure 27: Evolution of relays appearing on block lists.

and co-existence of spam-related URLs on block lists implies that the benefits for spam producers to continue including spam-related URLs in their messages may outweigh the costs of being identified and filtered by the block list spamicity tests.

One possible explanation for this phenomenon involves a cost/benefit analysis from the spam producer's point of view. It is obvious that having a convenient URL directly pointing to the spam producer's Web site is very valuable. If the spamicity test is able to completely and immediately filter out all such spam messages, this strain of spam would probably become extinct. Since we assume the block list coverage is very good, the main question is the length of the time lag between the creation of a spam-related URL and its detection and inclusion on a block list. Despite the presence of effective collaborators, it is reasonable to assume a non-negligible time lag (e.g., on the order of hours or days) exists between the creation of a new URL and its discovery and inclusion on a block list. This time lag could be a fundamental limitation of the collaborative filtering approach, which is based on effective human participation.

Another analogous collaborative approach is DNS block lists. Unlike the URL block lists described above, DNS block lists attempt to filter messages based on the servers that were used to deliver the messages. Figure 27 shows the evolution of the number of spam messages that were delivered through at least one relay that appears on a DNS block list. Three block lists are represented: The Distributed Sender Blackhole List (DSBL), The Spam and Open Relay Blocking System (SORBS), and NJABL.ORG. We observe both

growth (probably due to the stabilization process described above for collaborative filters in general) and decline of this spamicity test in Figure 27. However, as of December 2005, the spam messages that pass through at least one of the relays on these lists did not become extinct (at around 20%). We believe the time lag explanation above also applies here.

5.4 Related Work

Previous studies that investigated the evolution of spam were primarily concerned with the content of spam messages. Fawcett [50] discovered a few interesting spam trends that occurred in 2002. In his study, he found a great deal of variation in the traffic patterns of spam and legitimate email messages, and using these variations, he illustrated the time variation of the class prior p(spam). He also investigated the evolution of spam message content over time, finding a number of complex trends. Specifically, he found that spam terms (i.e., words that appear in spam messages) fall into a combination of three categories: constant, periodic, and episodic occurrences. Finally, Fawcett mentioned the early stages of the "spam arms race," primarily focusing on simple techniques that were created to defeat keyword filters.

On two separate occasions [22, 23], Brightmail published statistics about the evolution of spam traffic and spam content. In the first set of statistics [22], they showed that from January 2003 through December 2003, the percentage of all email that was spam grew from 42% to 58%. Additionally, in December 2003, they found that most spam messages were categorized as PRODUCTS (21%) and ADULT (18%). In the second set of statistics [23], they showed that from January 2004 through March 2004, the percentage of all email that was spam grew from 60% to 63%. Additionally, in March 2004, they found that most spam messages were categorized as PRODUCTS (25%) and FINANCIAL (20%). In the middle of 2005, Sophos also released statistics that provided spam content categorizations. Specifically, they found that from January 2005 through June 2005, Medication/pills was the top spam category (41.4% of all spam messages during that period), followed by Mortgage (11.1% of all spam messages during that period).

Hulten et al. [85] "hand-examined" 200 spam messages from a one month period in 2003

and 1000 spam messages from a one month period in 2004. The purpose of this examination was to identify the types of products being promoted and the types of exploits being used by spam messages. Their main observations were that non-graphic porn/sex content was the most prevalent spam category and that "text chaff" (i.e., textual obfuscation techniques) was the most prevalent exploit in their data.

Our study differs from the previous work on spam evolution in several ways. First, we study the evolution of spam construction techniques in spam messages, instead of spam content. Second, our study uses large corpora (over 1.4 million spam messages over a three year period) to produce concrete and clear evidence of evolution. Third, we focus on two clear trends (extinction and co-existence) that give us a quantitatively supported evaluation of spamicity test effectiveness in "killing" spam messages (either completely for the extinction group or partially for the co-existence group).

5.5 Summary

In this chapter, we studied the evolution of spam, focusing on a trend analysis of spam construction and filtering techniques. The study used over 1.4 million spam messages that were collected from SpamArchive between January 2003 and January 2006. The spam constructions and filtering techniques were adopted from the spamicity tests found in SpamAssassin 3.1.0. The study ran all of the messages through all of the spamicity tests, and it plotted the percentage of messages for which the test result was positive through the three-year period.

We consider two trends in this study: the spam construction techniques that became "extinct" (zero or near zero spam messages for that spamicity test) and the spam construction techniques that survived and co-exist with a well-defined spamicity test. We divide the explanations of these trends into three groups of spamicity tests: significant environmental changes, individual filtering, and collaborative filtering. Extinction of a spam construction technique means complete filter effectiveness (e.g., individual filtering of HTML-based obfuscation techniques) or environmental changes (e.g., the elimination of USERPASS functionality in browsers). In contrast, co-existence indicates the existence of

concrete limitations in the spam filters. Identified examples include forged Outlook "X-Mailer" headers and illegal characters in "Subject" headers for individual filters and block lists for collaborative filtering.

CHAPTER VI

DEFENDING CLASSIFIERS AGAINST CAMOUFLAGED EMAIL SPAM

Spam email has become a costly and pervasive problem. In response, the Messaging Anti-Abuse Working Group (MAAWG), formed by major Internet Service Providers, began tagging and blocking spam messages in the backbone of the Internet. According to the 2006 First Quarter Report of MAAWG, close to 370 billion emails were tagged or blocked as spam in the backbone, compared to 90 billion emails that were considered legitimate and delivered to their destinations. Despite this partial success, we showed in the previous chapter that none of the current defensive techniques against spam are effective over a long period of time due to the continuous adaptation of spammers. As an example of successful spammer adaptation, anecdotal evidence suggests that many of the spam messages that are reaching their destinations contain *camouflaged* content (i.e., legitimate tokens that are unrelated to the spam message and used to mask spam content). This camouflage follows a history of spammer adaptation, which began with the many misspellings of VIAGRA and has become increasingly sophisticated over time.

Camouflaged spam messages were originally created in response to statistical learning filters [6, 7, 8, 9, 46, 70, 117, 127, 151, 156], which analyze the textual content of emails to distinguish spam from legitimate messages. Due to the initial success of learning filters (against spam with little to no camouflage), they have been incorporated into major mail clients (e.g., Microsoft Outlook and Eudora) as well as many mail servers. One of the most important advantages of statistical filters is their ability to learn the many variations of spam text (e.g., misspellings of VIAGRA) automatically from new spam messages. Unfortunately, as we found in Chapter 4, the increasing amount of camouflage in spam creates an instance of the *adversarial classification* problem, where spammers attempt to use legitimate tokens as camouflage to deceive these filters. Currently, most adversarial classification researchers

believe a general solution does not exist for this problem because the victims are always reacting to spammers' new camouflage (see Section 6.1 for more details), perpetuating a continual camouflage arms race.

Although both the history of the spam domain and the general adversarial classification field suggest a never ending camouflage arms race, the main contribution of this chapter is an approach to statistical learning filter design that helps to escape this arms race. Our method makes statistical learning filters resistant to randomized, camouflaged content by exploiting a careful assignment of weights to spam features (scores for the tokens associated with spam messages) and legitimate features (scores for the tokens associated with legitimate messages). The differentiated treatment of spam features and legitimate features is implemented by retaining a sharply different number of legitimate and spam features when training the learning filters and giving them appropriate weights. As a result, we are able to increase the sensitivity of learning filters to detect the spam tokens, even in the presence of camouflage. An intuitive explanation is based on the observation that "strong" spam tokens (also known as low-entropy features – e.g., misspellings of VIAGRA) are good indicators of spam, and as such, they should receive special attention because legitimate messages usually do not contain such low-entropy spam features.

The goal of spammers is to use camouflaged content to make their spam messages "look like" legitimate messages to text-based learning filters. Despite the presence of this camouflaged content, we argue that text analysis is a fundamental part of content filtering because legitimate emails are typically compromised of a non-trivial percentage of text. Thus, most of the messages that contain only images or URLs can be easily classified as spam because very few legitimate messages are dominated by images or URLs. Consequently, we conjecture that spammers will inevitably use a significant amount of camouflaged text to effectively emulate legitimate emails. This observation (and conjecture) suggests that our method to detect spam, despite camouflage, may have increasing relevance and impact as spammers continue to add a growing amount of camouflage to their spam messages.

Our experimental results show that the combination of a small number of legitimate features and a large number of spam features achieves two goals. First, the small number

of legitimate features is an effective strategy to reduce the influence of camouflage because an increased number of legitimate features in a camouflaged message will no longer significantly increase the probability of the message being legitimate. Second, by increasing the number of spam features, we can capture all of the strong spam features and find each spam message, even if it contains only a small number of spam features. In our experiments, this strategy restores the performance of our learning filters (Naïve Bayesian, SVM, and LogitBoost) against camouflage attacks, maintaining less than 10 percent false negatives and less than 2 percent false positives, even with one thousand legitimate tokens added to the camouflaged messages. More importantly, our method gives hope that camouflage attacks can be countered by carefully tuning learning filters.

The rest of the chapter is organized as follows. We outline spam defense background information and other related work in Section 9.4. In Section 6.3, we summarize our experimental setup and baseline filter performance. In section 6.4, we evaluate the effectiveness of camouflage attacks on learning filters. In Section 6.5, we evaluate the effectiveness of retraining the learning filters and the attack/retrain cycle. In Section 6.6, we show that by carefully tuning the learning filters, we can restore their effectiveness in distinguishing camouflaged spam from legitimate messages. Section 6.7 summarizes our results.

6.1 Background on the Spam Arms Race

Historically, spam producers and receivers (called victims) have been engaged in an arms race ever since spam was introduced. In round one of this arms race, the victims created keyword-based filters to distinguish spam from legitimate messages. Typically, keyword filters are able to find incriminating words in specific parts of messages (e.g., "Ecolife Company" as the sender or "online pharmacy" in the message body). More advanced filters can combine these criteria into sophisticated predicates. In response (round two), the spam producers adopted the misspelling attack, which is very effective against keyword filters because those filters depend on a victim's precise specification of what keywords are found in spam messages [151]. A major problem for the victims is that keyword filters need to be maintained manually, while misspellings can be easily and automatically produced by

software tools. Due to the misspelling attack, most victims have reduced their reliance on keyword filters, and as a result, spam producers are usually considered the winners of round two.

With the decline of keyword filters, round three began when victims adopted statistical learning filters, which are based on machine learning algorithms such as Naïve Bayes, Support Vector Machines (SVM), and LogitBoost. These statistical filters have a learning phase in which they associate words (also called tokens or features) from known spam messages with "spamicity" scores and words from legitimate messages with "legitimacy" scores. The learning filters automate the process of learning the misspellings used by spam producers, becoming capable of identifying spam with misspellings with minimal guidance by victims. These learning filters have been shown to be quite effective in distinguishing early spam from legitimate messages [6, 7, 8, 9, 46, 70, 117, 127, 151, 156].

In response to learning filters, spam producers entered round four by introducing camouflaged content into their spam messages and launching camouflage attacks. Typically, the camouflaged content consists of fragments of legitimate text or simply a list of commonly used words. These words, which are associated with legitimate messages, increase the legitimacy scores of the camouflaged spam messages and make them look more like legitimate email. In Chapter 4, we showed that filters trained without the knowledge of camouflaged messages have serious difficulties identifying those camouflaged messages as spam.

In response to camouflaged content, victims began round five by using refined learning, a process in which learning filters are trained with spam messages that contain camouflaged content. The refined learning filters acquire the ability to identify spam with camouflaged content, but they are only effective against the known camouflage used in the training. In response to this refined training, spam producers can use software tools to randomize the camouflaged content in round six. The randomized camouflage is able to confuse the refined filters because the filters have not seen it before (see Section 6.5.3). Ominously, this situation is analogous to round two, where automated randomization of misspellings defeated keyword filters, suggesting a potential defeat of learning filters in the spam arms race. Fortunately, in this chapter, we provide a solution to camouflage attacks, which

restores the upper hand in the spam arms race to the victims.

6.2 Related Work

The use of machine learning algorithms in spam filters is a well established idea. Previous research [6, 7, 9, 117, 127] has shown that the Naïve Bayes algorithm can be used to build very effective personalized spam filters. Similar results have also been obtained for Support Vector Machines [9, 46, 156] and Boosting algorithms [9, 61, 156].

In order to be successful, learning filters must identify features that distinguish legitimate instances from spam instances. As a result, these filters are vulnerable to attack by spam messages that possess a significant number of legitimate features. For example, Graham-Cumming [70] described an attack against an individual user's spam filter that involved inserting random dictionary words into spam messages. Then, Wittel and Wu [151] added commonly occurring English words to spam messages instead of random words. As another extension, we used portions of legitimate message to construct camouflaged messages in Chapter 4. Lowd and Meek [104] explored the effectiveness of both active and passive attacks with varying amounts of legitimate content added to attack spam messages. Through their evaluation, they conclude that the only remedy for filtering attack messages is through frequent retraining. More generally, there is an area of the machine learning field called adversarial classification. For example, Dalvi et al. [42] studied attacks against spam filters using game theoretic methods. They found that the iterations between attacking and retraining spam filters continue endlessly.

Our research in this chapter differs from the previous work in two main ways. First, the attack messages in our experiments are significantly more sophisticated, utilizing legitimate tokens from the learning filters to maximize the confusion in the filters. Second, instead of retraining spam filters, we describe a novel approach to design camouflage-resistant spam filters. To the best of our knowledge, this is the first published solution to the problem of endless iterations in the camouflage arms race.

A complementary alternative to the automated retraining of learning filters is human-driven collaborative filtering, available to large email service providers such as Hotmail and

Gmail. The service providers supply a user-controlled button that indicates a message is spam, as judged by that user. For such large groups of users, several clicks from independent users may be a strong indication of the message being spam, allowing efficient filtering before the other users have seen the message. The advantage of collaborative filtering is the sharing of retraining work through collaboration. However, it is not available for smaller email service providers. Furthermore, although collaborative filtering appears to have been quite effective, it is vulnerable in two ways. First, it contains an inherent delay between the appearance of new camouflaged content and the actual retraining process. During this period, the camouflage attack is very effective (see Section 6.4). Second, it is vulnerable in a way similar to adversarial classification since spam producers can infiltrate the collaborative filtering process by providing confusing judgment (e.g., clicking on the spam button on legitimate messages to increase the false positive rates of the collaboratively trained filter). We see collaborative filtering as a technique that can improve the efficiency of individual filters, while not replacing them.

6.3 Baseline Evaluation of Learning Filters

In this section, we summarize the settings for the experiments described in Sections 6.4, 6.5, 6.5.2, 6.5.3, and 6.6. Although the evaluation of spam filtering is a well known process [6, 7, 8, 9, 46, 117, 127, 156], we outline two issues in this section. First, we discuss the processing of each message, including feature selection and header selection to minimize corpus-specific bias. Second, we summarize the selection of large corpora (hundreds of thousands of messages) in our experiments and include baseline filtering results (to be compared to the attack results in subsequent sections). For readers unfamiliar with machine learning, we have included a short explanation of the basics of statistical learning filters in the Appendix.

6.3.1 Training and Test Corpora

It is a standard practice in machine-learning-based text classification research [103, 105, 114, 125] to use large corpora in evaluation experiments. In Chapter 3, we presented quantitative arguments in favor of using large samples from large corpora (on the order

of hundreds of thousands messages) in experimental evaluations of spam filters. This is in contrast to previous evaluation experiments involving spam filters [6, 7, 8, 9, 46, 70, 117, 127, 151, 156] that have typically used only very small samples from small corpora (on the order of a few thousand messages). The experiments in that chapter show that small corpora lead to significantly less predictable results than large corpora. According to our own recommendations, we use large corpora in the research reported in this chapter.

For spam message training, we collected $750K$ spam messages from the publicly available spam corpora maintained by SpamArchive[1] and SpamAssasin[2]. For legitimate message training, we started with the Enron corpus[3] [95]. However, in our experiments with this corpus, we discovered a number of messages that appeared to be spam. To reduce the noise in the data set, we used a Naïve Bayesian spam filter (trained with $5K$ randomly selected spam messages and $5K$ randomly selected legitimate messages) to classify the Enron messages. Then, we discarded the messages that were classified as spam. After this cleaning process, the remaining $475K$ legitimate Enron messages were used as legitimate messages in our experiments.

All of the collected messages were tokenized using a software tool called SpamProbe [25]. Each message is represented as a "set of words" (i.e., the set of tokens in the message). Using this representation, each message is represented as a feature vector $\mathbf{f}$ of n features: $< f_1, f_2, \ldots, f_n >$. All of the features are Boolean; thus, if $f_i = 1$, the feature is present in a given instance; otherwise, the feature is absent. This is a commonly used representation that has been validated by previous experiments [10, 48, 129].

6.3.2 Message Processing

A concern in the processing of messages is the presence of systematic bias in the corpora chosen for training. For example, in previous work, email headers were associated with improved spam filtering performance [156]. Our experiments confirmed the improvements, which appear to have been caused by biases in the corpora. An example is the

[1]SpamArchive's spam corpora can be found at ftp://spamarchive.org/pub/archives/.
[2]SpamAssasin's spam and legitimate corpora can be found at http://spamassassin.org/publiccorpus/.
[3]The Enron corpus can be found at http://www-2.cs.cmu.edu/ enron/.

"Message-ID" header in the Enron corpus because many message headers contain "JavaMail.evans@thyme" as a substring. Consequently, that substring artificially increases the legitimacy scores of any message containing it. Conversely, the "Received" header is present in 85% of the messages in our SpamArchive corpus, but it is absent from the messages in the Enron corpus. To remove the systematic biases due to these header types, we use messages with only two kinds of headers: "Subject" and "Content-Type."

6.3.3 Effectiveness of Baseline Filters

In our experiments, we randomly selected a training set of $10K$ messages ($5K$ legitimate messages and $5K$ spam messages) and a test set of $10K$ messages ($5K$ legitimate messages and $5K$ spam messages). In Chapter 3, these settings found a representative sample and possess sufficient variety to generate reproducible results. To establish a baseline of comparison for the attack experiments described in subsequent sections, we initially trained our spam filters (Naïve Bayes, SVM, and LogitBoost) with a $10K$ message training set using three different combinations of retained features during the training phase: 25, 100, and 300 legitimate features and an equal number of spam features. These feature were retained using the Information Gain feature selection algorithm.

The results of the baseline filter evaluation are summarized in Table 17, which shows the accuracy (in terms of false negative rates and false positive rates) of the filters being evaluated (Naïve Bayes, SVM, and LogitBoost). We see that SVM and LogitBoost generate low false negative rates (low single digit percentages below 2%), with slight improvements as the number of training features increases. The performance of Naïve Bayes remains at or below 20% false negatives for all three training settings. Due to their tuning, the filters are conservative with respect to false positives. The false positive rates for SVM and LogitBoost vary from less than 2% for 25 legitimate and 25 spam training features to around 0.5% for 300 legitimate and 300 spam training features. Naïve Bayes also exhibits very low false positive rates (0.05% for 25 legitimate and 25 spam training features and 0.02% for 300 legitimate and 300 spam training features). Due to the very low false positive rates, we omit the false positive figures in the chapter, except for Figure 34, where we show

Table 17: Baseline performance of the filters.

Filter	False Negative Rate	False Positive Rate
25 Legitimate and 25 Spam Features		
LogitBoost	0.0191	0.0164
Naïve Bayes	0.1925	0.0005
SVM	0.0190	0.0169
100 Legitimate and 100 Spam Features		
LogitBoost	0.0133	0.0048
Naïve Bayes	0.2043	0.0003
SVM	0.0145	0.0052
300 Legitimate and 300 Spam Features		
LogitBoost	0.0134	0.0036
Naïve Bayes	0.1777	0.0002
SVM	0.0126	0.0056

the equally low false positive rates of our new filter design.

6.4 Evaluation of Camouflage Attacks

In this section, we study the camouflage attacks in round 4 of the spam arms race outlined in Section 6.1. We describe a camouflage attack strategy against learning filters and evaluate three popular machine learning algorithms used in spam filtering: Naïve Bayes, SVM, and LogitBoost. We assume that spam producers have complete knowledge of both the filters we use and the corpora we use for training. There are two reasons for this worst-case assumption in adversarial classification. First, the filters and the corpora used are public knowledge, and as a result, spam producers have easy access to them. Second, our results do not depend on any secret knowledge that is hidden from the adversaries, making the results generally applicable to several filters and corpora (as described below).

6.4.1 Construction of Camouflaged Spam Messages

In Chapter 4, we found that a simple method of creating camouflaged spam messages involves appending a fragment of a legitimate message to the end of a spam message. The embedded legitimate content increases the legitimacy score of camouflaged messages. A technically more sophisticated and effective method for creating camouflaged messages

involves adding only *legitimate tokens* to spam messages. Intuitively, legitimate tokens are tokens that appear often in legitimate messages but rarely in spam messages during the training phase.

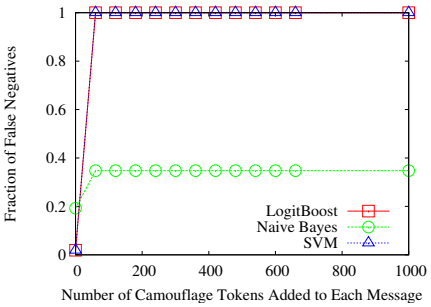
More precisely, legitimate tokens are characterized by high scores in three quantitative properties: information content, probability of occurring in legitimate messages, and probability of occurring in randomly chosen training sets. They are the most effective tokens used as camouflage for three reasons. First, learning filters only use those tokens that are selected by their feature selection scheme (Section 6.3.2) for classification. Our feature selection relies on Information Gain; thus, tokens used for camouflage should have high Information Gain scores to influence the filters. Second, the legitimacy score of a token T is defined as the probability of that token occurring in a legitimate message:

$$Legitimacy(T) = \frac{N_{legitimate}(T)}{N_{legitimate}(T)+N_{spam}(T)},$$

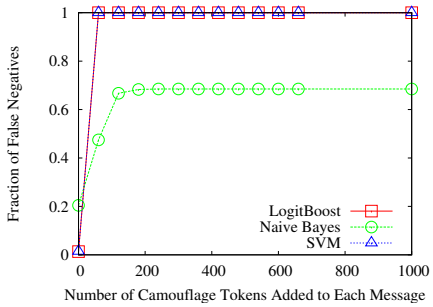
where $N_{legitimate}(T)$ is the number of legitimate messages in which T occurs and $N_{spam}(T)$ is the number of spam messages that contain T. Third, a successful camouflage token must be present in the training sets of filters; thus, a high probability of occurrence in randomly selected training sets (common words) is favored over rare words. To determine the set of legitimate tokens for a given collection of large corpora, we run a number of spam evaluation experiments (as outlined in Section 2.2). By extracting the legitimate tokens with high scores in the properties described above, we can find the most effective tokens for a camouflage attack (described in the following sections).

6.4.2 Effectiveness of Camouflage Attacks

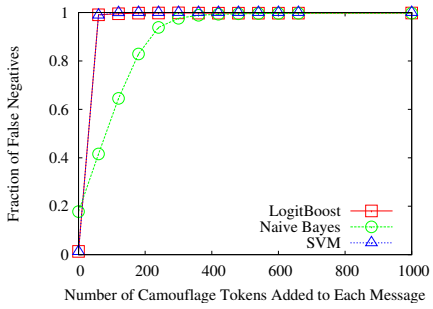
Our next set of experiments evaluates the effectiveness of legitimate tokens in a camouflaged message for confusing learning filters that were trained in the baseline experiments outlined in Section 6.3.3. The legitimate test messages are the same as in the baseline experiments ($5K$ legitimate messages), but the $5K$ camouflaged spam messages are constructed by adding legitimate tokens to the original spam messages. To quantify the influence of legitimate tokens, we gradually increased the number of added legitimate tokens from 0 to 660, in increments of 60, with a final data point at 1000 added tokens. Each experiment was



(a) 25 legitimate and 25 spam features



(b) 100 legitimate and 100 spam features



(c) 300 legitimate and 300 spam features

Figure 28: Performance of baseline filters under attack. In (a), the filters are built using 25 legitimate and 25 spam features. In (b), the filters are built using 100 legitimate and 100 spam features. In (c), the filters are built using 300 legitimate and 300 spam features.

repeated 10 times, and the averages are reported in Figure 28 in terms of false negatives (spam messages that have not been recognized as such). We have omitted figures for the standard deviation and coefficient of variance values because they are consistently low (i.e., the highest coefficients of variance are rarely larger than 5%), indicating a great deal of precision for each of our experimental runs. Also, since the camouflage attack is aimed at increasing false negative rates, we omit the graphs for false positives in this and following evaluation sections. The number of false positives remains very small in these experiments (one example will be shown in Figure 34).

Figure 28(c) shows the rapidly increasing success of camouflage attacks for the Naïve Bayes, SVM, and LogitBoost filters that are trained with 300 legitimate and 300 spam features. We can see that in the baseline case (with 0 attack tokens), all of the filters have

low false negative rates. However, the situation changes quickly as camouflage tokens are added into the attack messages. The LogitBoost and SVM filters quickly identified the legitimate tokens and classified the camouflaged messages as legitimate (false negatives), starting from 60 attack tokens. The Naïve Bayes filter was slightly more robust, exhibiting a gradually increasing false negative rate that converged to 100% at 360 tokens.

This trend becomes more pronounced as the number of features used in the training is decreased to 100 legitimate and 100 spam features in Figure 28(b) and 25 each in Figure 28(a). As the number of training features decreases, the SVM and LogitBoost filters remain vulnerable to camouflage attacks, but the Naïve Bayes filter shows an increasing "resistance" to camouflage attacks. Using 100 legitimate and 100 spam features, the Naïve Bayes filter is confused only about 70% of the time, and at 25 legitimate and 25 spam features, the confusion rate is reduced to less than 40%.

To explain this apparent camouflage resistance at lower training feature set sizes, we observe that in many machine learning experiments, the sensitivity and performance of filters are increased when we increase the training feature set size of the filters. We conjecture that such increased sensitivity might actually make the filters more vulnerable to camouflage attacks since they are identifying the camouflage and incorporating it into the classification decision. At 25 legitimate and 25 spam training features, the Naïve Bayes filter appears to be much more difficult to confuse, regardless of the number of legitimate tokens added (the circle line in Figure 28(a) is flat).

This intuitive explanation for the camouflage resistance of the Naïve Bayes filter is supported by an analysis of the probabilistic scores assigned to the spam messages. Comparing the baseline and the attack test sets, we see that using 25 legitimate and 25 spam features in training reduces the score changes due to the addition of camouflage. As a concrete example, a spam message M_{spam} was assigned the following unnormalized probabilistic scores: $P_{spam}(M_{spam}) = 3.23 * 10^{-3}$ and $P_{legitimate}(M_{spam}) = 8.51 * 10^{-27}$. Thus, the filter decided that M_{spam} was spam. During the camouflage attack test, 600 legitimate tokens were added to M_{spam}, producing M_{camou}. The filter assigned new scores: $P_{spam}(M_{camou}) = 2.91 * 10^{-21}$ and $P_{legitimate}(M_{camou}) = 2.83 * 10^{-25}$. Thus, we see a significant decrease in the spam score

and an increase in the legitimacy score of M_{camou}. However, the changes were insufficient to cause the filter to switch the decision from spam to legitimate, and M_{camou} remains classified as spam. This observation will be explored in depth when we describe our new filter design (Section 6.6).

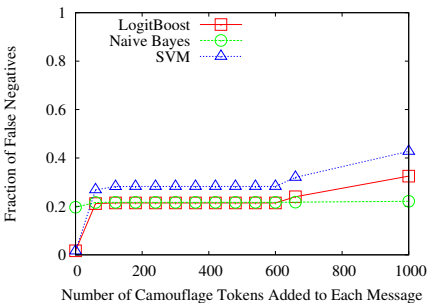
6.5 Evaluation of Retrained Filter Defense

6.5.1 Effectiveness of Retrained Filters

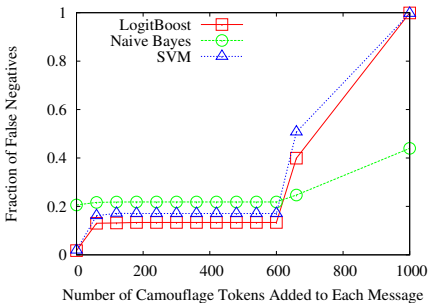
In the previous section, we showed that baseline filters are easily attacked by camouflaged messages. In response (Round 5 of the spam arms race outlined in Section 6.1), typical email managers and users resort to retraining their filters to counteract the effects of camouflage. During retraining, filters are explicitly trained to classify camouflaged messages as spam. This strategy is non-trivial in practice since distinguishing the camouflaged messages is usually a human activity (e.g., individual or collaborative filtering), and tools such as learning filters have difficulties with this task (as shown in the previous section). The goal of this section is to study the effectiveness of retraining, not to improve the performance or reduce the cost of the retraining strategy.

To evaluate the effectiveness of retraining, we added 600 legitimate camouflage tokens to a percentage of the spam messages in the filters' training sets, and then, we repeated the first experiment in Section 6.4.2. After numerous validation experiments, we found that the filters' were most effective when we added the camouflage tokens to 20% of the training spam messages. Figure 29 presents the average false negative rates for the retrained filters. In Figure 29(a), we show the results from filters trained with 25 legitimate and 25 spam features. The Naïve Bayes, SVM, and LogitBoost filters all consistently have false negative rates between 20% and 30%. This is a significant improvement over the results from the previous round in Figure 28(a).

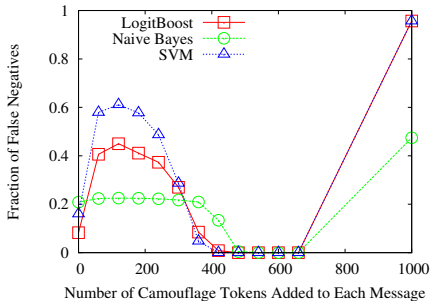
When the number of features in the filter training is increased to 100 legitimate and 100 spam features, Figure 29(b) shows that the false negative rates for the SVM and LogitBoost filters improve to 21% and 13% (respectively) for attack messages that contain up to 600 camouflage tokens. The situation becomes more complex when the number of retained



(a) 25 legitimate and 25 spam features



(b) 100 legitimate and 100 spam features



(c) 300 legitimate and 300 spam features

Figure 29: Performance of retrained filters. In (a), the filters are built using 25 legitimate and 25 spam features. In (b), the filters are built using 100 legitimate and 100 spam features. In (c), the filters are built using 300 legitimate and 300 spam features.

features is increased to 300 legitimate and 300 spam features. Figure 29(c) shows that the SVM and LogitBoost filters successfully classify all camouflaged messages containing between 420 and 660 camouflage tokens. However, the filters show different performance losses and gains for attack messages containing between 60 and 360 camouflage tokens. Although the Naïve Bayes filter exhibits reasonably stable performance, the performance of the SVM and LogitBoost filters is characterized by a bell-shaped curve, with the highest false negative rates around 120 legitimate tokens. This bell-shaped curve is an interesting open research question. The false positive figures are omitted since the rates continue to be very low (i.e., the same range as those listed in Table 17).

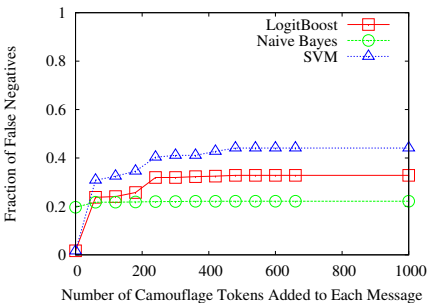
In Figures 29(b) and 29(c), the filter performance degrades noticeably for attack messages with 660 and 1000 camouflage tokens. This degradation is due to a limitation in

the retraining process: only 600 legitimate tokens were used as camouflage in the spam training set. Therefore, when more than 600 camouflage tokens appear in a test message, some of the tokens are necessarily new to the filter (i.e., they were not seen by the filters during the retraining process). The new tokens illustrate the vulnerability of the retrained filters, which repeats the situation discussed in Section 6.4. This situation indicates the continuation of the camouflage arms race with the retraining approach (as discussed in the next section).

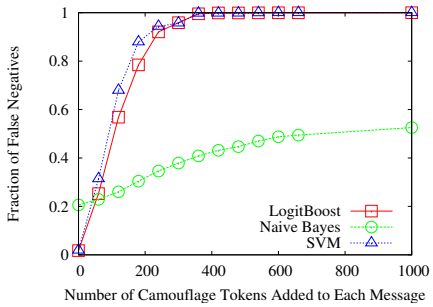
6.5.2 Retrained Filters Under Camouflage Attack

The attack process used against the retrained filters is the same as the attack process used against the baseline filters described in Section 6.4. The new problem faced by the attacker is finding appropriate legitimate tokens to construct effective camouflaged messages to bypass the retrained filters. Perhaps to the advantage of spam producers, the solution to this new problem is very similar to the selection algorithm used to obtain the original camouflage tokens (described in Section 6.4.1). First, tokens with legitimacy scores greater than 0.5 are extracted from the retraining process. From these legitimate tokens, the ones with the highest Information Gain are selected for the next iteration of camouflage attack. This process naturally eliminates the legitimate tokens used in the previous iteration since the Information Gain of those tokens is reduced by their presence in both legitimate and spam messages. The selection process is repeated 100 times to determine the set of tokens that have a statistically high probability of occurring in any randomly selected training set. Those tokens are then used for constructing new camouflaged messages for the next iteration of attack experiments, which use the retrained filters from Section 6.5.

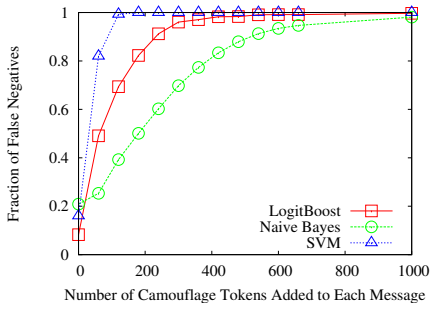
Figure 30 shows the results of attacking the retrained filters with new camouflage. Figure 30(c) shows accuracy results that are similar to Figure 28(c), starting from near 0 false negatives (for the SVM and LogitBoost filters) before the attack and rising to almost 100% false negatives when under attack. Since those figures have the same experimental settings (300 legitimate and 300 spam training features), they show that retrained filters lose their accuracy in a similar way. The SVM filter is the most sensitive to camouflaged content,



(a) 25 legitimate and 25 spam features



(b) 100 legitimate and 100 spam features



(c) 300 legitimate and 300 spam features

Figure 30: Performance of retrained filters under attack. In (a), the filters are built using 25 legitimate and 25 spam features. In (b), the filters are built using 100 legitimate and 100 spam features. In (c), the filters are built using 300 legitimate and 300 spam features.

reaching 100% false negatives with 120 new camouflage tokens. The LogitBoost filter is also quite sensitive to the camouflaged content, reaching 100% false negatives with 420 new tokens. The accuracy of Naïve Bayes degrades the slowest, reaching only 90% false negatives with 600 new tokens.

While Figures 30(c) and 30(b) show the same trends, Figure 30(b) exhibits a slower accuracy degradation. Figure 30(b) also exhibits a slower loss of accuracy than Figure 28(b), which also has filters trained with 100 legitimate and 100 spam features. This trend is particularly visible with the Naïve Bayes curve in Figure 30(b), which remains at less than 50% false negatives, compared to the Naïve Bayes curve in Figure 28(b), which flattens at about 70% false negatives.

Continuing this trend, Figure 30(a) is perhaps the most interesting figure because it

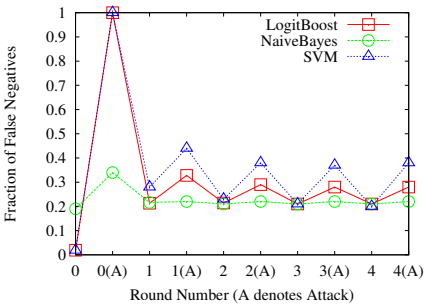
resembles Figure 29(a) more than Figure 28(a). Specifically, for the SVM and LogitBoost filters, Figure 30(a) shows that a small number of training features gives the filters a similar camouflage resistance that we observed with the Naïve Bayes filter in Figures 28(a) and 30(b). This resistance to camouflage attacks for a small number of training features will be explored in Section 6.6.

6.5.3 Iterative Retraining in the Camouflage Arms Race

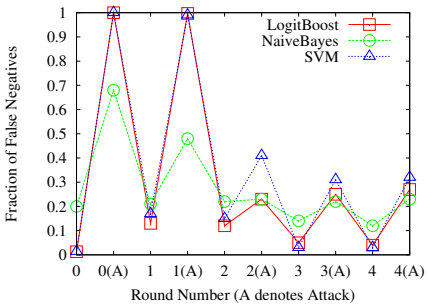
Section 6.4 shows that learning filters are vulnerable to camouflage attacks (Round 4 of the spam arms race). Sections 6.5.1 and 6.5.2 show that while the retraining strategy is effective for distinguishing the camouflaged messages that have been seen previously (Round 5), the retrained filter remains vulnerable to the same camouflage attack using legitimate tokens that have not been used in an attack (Round 6). An important question is whether this cycle in the camouflage arms race will come to an end as we repeat the retraining and renewed camouflage attacks. The following experiments suggest that such a cycle does not seem to end quickly when new camouflage is chosen as described in the previous section.

Let us denote the baseline filter as F_0 and the set of legitimate tokens associated with F_0 as L_0 (i.e., the attack token set for F_0). The retrained filter is denoted by F_1. From F_1, we extract a new set of legitimate tokens L_1. As the iteration process continues, for each retrained filter F_i, we have its associated legitimate tokens L_i. In Figure 31, each filter evaluation experiment is denoted by an iteration number on the x-axis, starting from 0 (the baseline). In each iteration, the retrained filter evaluation is followed by the evaluation of a renewed camouflage attack, denoted by the iteration number concatenated with (A). Each renewed camouflage attack contains a set of 600 legitimate tokens that are associated with the corresponding retrained filter.

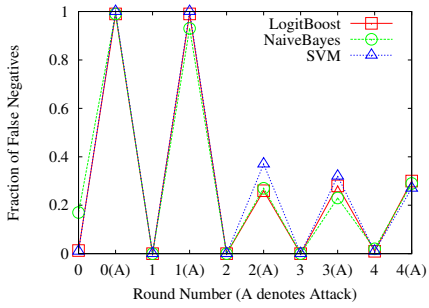
In Figure 31, each retrained filter F_i appears as a valley (i.e., it has a low average false negative rate), and its vulnerability to camouflage attacks using L_i appears as a peak. The general trend is the same for all iterations: declining but sustained average false negative rates under renewed camouflage attack, for each iteration. Figures 31(a), 31(b), and 31(c) show subtle differences among the three filter training feature size settings. While a small



(a) 25 legitimate and 25 spam features



(b) 100 legitimate and 100 spam features



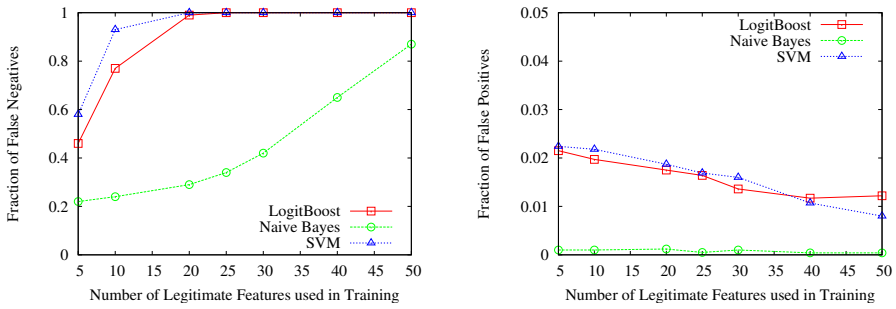
(c) 300 legitimate and 300 spam features

Figure 31: Illustration of the camouflage attack/retraining cycle. In (a), the filters are built using 25 legitimate and 25 spam features. In (b), the filters are built using 100 legitimate and 100 spam features. In (c), the filters are built using 300 legitimate and 300 spam features.

number of features (25 legitimate and 25 spam) shows good camouflage resistance in Figure 31(a) with relatively low peaks, the same filters are unable to achieve the normally high accuracy of learning spam filters. The figure shows relatively high valleys, compared to the small single digit false negative rates in Table 17. Filters in Figures 31(b) and 31(c) achieve a high level of accuracy in their valleys (comparable to Table 17), but the attacks are able to successfully maintain around 30% average false negatives for the peaks.

6.6 Learning Filters Resisting Camouflage Attacks

With the apparently continuing camouflage attack/retraining cycle shown in Figure 31, we have reached the main question of this chapter: "Is it possible to stop or circumvent the



(a) (b)

Figure 32: Attack-resistance of baseline filters with a varying number of legitimate features used in training.

camouflage attack/retraining cycle?" We give a positive answer in this section.

6.6.1 Analysis of Camouflage Attacks and Filter Tuning

To find a way to circumvent the camouflage attack/retraining cycle, we return to the resistance to attacks shown in Figures 28(a), 30(a), and 30(b). The performance of the Naïve Bayes filter in these figures shows that training with a small number of legitimate features seems to increase the resistance against camouflage attacks. We investigated this possibility with a more detailed study of filter performance by varying the number of legitimate features used in training. As an illustration, Figures 32(a) and 32(b) show the changes in filter performance for a specific case (camouflaged spam with 600 attack tokens added), while we vary the number of legitimate features used in filter training between the range of 5 and 50, combined with a fixed spam feature set size of 25. Figure 32(a) shows that the filters' ability to successfully identify camouflaged messages degrades steadily (i.e., the false negative rates rise) as the number of legitimate features used in training increases, particularly for Naïve Bayes in the tested range. Figure 32(b) shows that the filters' ability to successfully identify legitimate messages increases slightly (i.e., the false positive rates decrease) as the number of retained legitimate features increases. Thus, the most effective approach involves retaining the smallest number of legitimate features that generates tolerable false positive rates.

Although Figure 32 shows the results of a simple experiment, it represents a significant departure from classic spam learning filters, which are typically trained with an equal number of legitimate and spam features. This is due to an implicit assumption that the contributions of each token (either spam or legitimate) would be the same when the filter processes a message. The camouflage attack forces us to reconsider this assumption since the camouflage tokens are now being counted on both sides: the legitimacy score (before the retraining) and the spamicity score (after retraining).

The observation of apparent attack resistance shown by a small number of legitimate training features may be explained in two ways. First, camouflage attacks depend on a sufficiently high legitimacy score. Thus, using a small number of legitimate features in the filter training will reduce the sensitivity to legitimate tokens by lowering the overall legitimacy scores of the messages being tested. Second, spam messages (whether camouflaged or not) typically contain a non-zero number of spam features. With a high number of spam training features, we can train the filter to have high spam sensitivity in detecting spam. Next, we show that the combination of high sensitivity to spam and low sensitivity to camouflage is the right combination to resist camouflage attacks.

6.6.2 Evaluation of Camouflage Attack Resistance

In this section, we describe and evaluate a new spam filter design capable of handling camouflaged messages with minimal retraining. We describe a quantitative analysis of the rationale for the design, and then, we summarize an experimental evaluation of that design. First, we establish the terminology used to describe the spam classification process.

- The spam filter training uses two feature sets: the legitimate feature set denoted by L and the spam feature set denoted by S.
- A camouflaged attack message N contains a set of original spam tokens denoted by N_S plus a set of appended camouflage legitimate tokens denoted by N_L.
- The set of legitimate tokens in N that are identifiable by the filter is $L \cap N_L$, and the corresponding set of identifiable spam tokens is $S \cap N_S$.

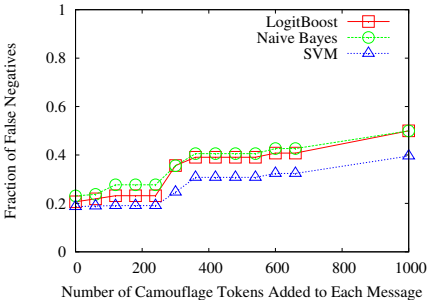
- The learning filter will classify N as spam if the spamicity score calculated from $S \cap N_S$ is greater than the legitimacy score calculated from $L \cap N_L$.

To achieve the combination of high sensitivity to spam and low sensitivity to camouflage, we divide the problem into two parts. The first goal of the new filter design is to decrease the sensitivity of filters towards legitimate (and camouflage) tokens. The legitimacy score is lower when the size of $L \cap N_L$ is smaller, and this is achieved by minimizing the size of L during training since N_L is under the control of the spam producer. The second goal of the new filter design is to increase the sensitivity of filters towards spam tokens. The spamicity score is higher when the size of $S \cap N_S$ is larger, and this is achieved by maximizing the size of S during training since N_S is under the control of the spam producer.

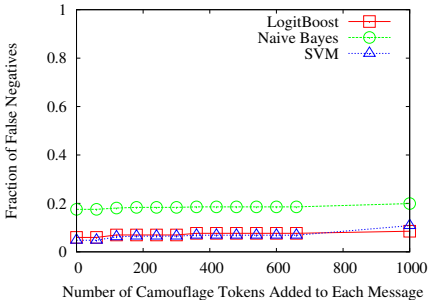
The combination of minimizing L and maximizing S during training is our basic strategy for increasing the filters' ability to detect camouflaged attack messages and achieve low false negative rates. In addition, if the legitimate messages do not contain spam tokens, this combination should be able to identify the legitimate messages and maintain low false positives. Now, we consider improvements on the basic design.

In addition to minimizing L, a second strategy to further reduce the size of $L \cap N_L$ is to choose L in a manner that makes it more difficult for spam producers to guess the content of L. If the camouflage tokens used by spam producers are outside of L, they become ineffective attack tokens since they do not contribute to the legitimacy score of the attack message N. In our notation, $L \cap N_L$ becomes empty. Unfortunately, classic feature selection algorithms such as Information Gain (Section 2.2) could make L easily reproducible by spam producers, facilitating their ability to guess L. Instead, we can randomly select the content of L from a large set of candidate legitimate features (e.g., features that exceed a threshold of Information Gain during feature selection). The randomization of L should reduce the intersection between N_L and L.

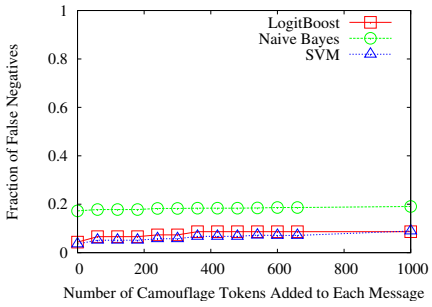
Finally, we need to consider another attack strategy that increases the size of $L \cap N_L$: sending an entire dictionary as the camouflage. This technique would guarantee that $L \cap N_L = L$, neutralizing the randomization of L described above. To prevent dictionary attacks, we can bound N_L by bounding the maximum size of the legitimate messages



(a) 25 legitimate and 1000 spam features



(b) 25 legitimate and 6000 spam features



(c) 25 legitimate and 9000 spam features

Figure 33: Attack-resistant spam filters. In (a), the filters are built using 25 legitimate and 1000 spam features. In (b), the filters are built using 25 legitimate and 6000 spam features. In (c), the filters are built using 25 legitimate and 9000 spam features. All features are randomly selected.

(excluding attachments) we consider with our approach. From our large legitimate collections (e.g., Enron) we have found empirically that less than 0.01% of all legitimate messages contain more than 600 tokens (excluding attachments). Consequently, we can give special treatment and filtering to messages with more than 600 tokens (e.g., individual or collaborative filtering techniques).

We summarize the above discussion with four heuristics for the design of camouflage resistant spam filters. The first three minimize the legitimacy score by decreasing filter sensitivity to legitimate features, and the last one maximizes the spamicity score by increasing filter sensitivity to spam features.

1. Minimize the number of legitimate features L that are used for training.

2. Randomize the content of L to reduce successful guesses by spam producers.

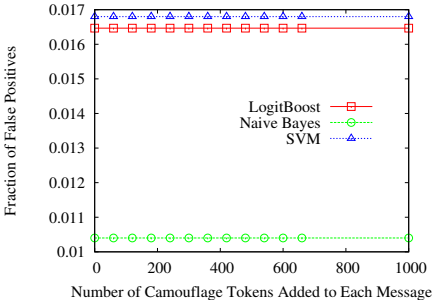
3. Limit the size of legitimate email messages handled by this approach.

4. Maximize the number of spam features S that are used for training.

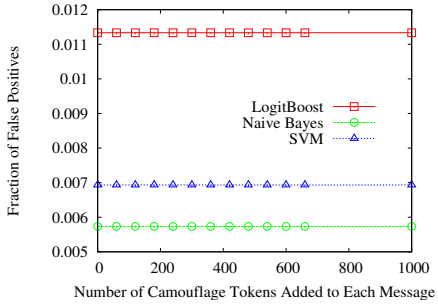
Figure 33 shows these heuristics to be effective against camouflage attacks with up to 1000 camouflage tokens. In this experiment, we did not bound the maximum size of legitimate messages (i.e., Heuristic 3) to ensure that the results are easy to compare with the results from previous experiments. Determining the effect of this additional heuristic is still an open research topic. In Figure 33(a), we trained the filters with 25 legitimate features and $1K$ spam features. We chose 25 legitimate features because that setting provided the best trade-off between false negative and false positive performance in Figure 32. Compared to Figure 28(a), this figure shows dramatic improvements for the SVM and LogitBoost filters, from almost 100% false negatives down to between 20% and 40% false negatives for up to 600 camouflage attack tokens. The performance of the Naïve Bayes filter is mixed, with some improvements in the 60 to 360 camouflage attack token range but comparable or higher false negatives for more than 360 attack tokens. Of the previous experiments, perhaps Figure 33(a) resembles Figure 30(a) the most, showing that the increase in the number of training spam features from 25 to 1000 may be insufficient.

By increasing the number of training spam features from $1K$ to $6K$ in Figure 33(b), we see significant improvements for the SVM and LogitBoost filters, which now achieve less than 10% false negatives. The Naïve Bayes filter also improves, generating less than 20% false negatives. In addition to improvements over Figure 33(a), Figure 33(b) is also clearly better than Figure 30(a). Perhaps more interesting is the fact that Figure 33(b) shows performance that is better than the retrained filters in Figure 29(b), particularly for large numbers of camouflage attack tokens (more than 600).

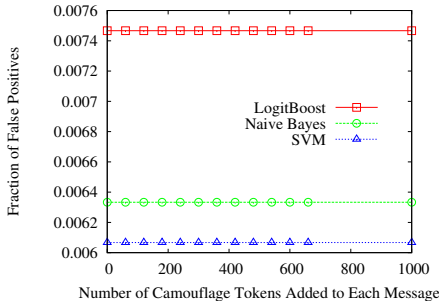
By increasing the number of training spam tokens to $9K$ in Figure 33(c), we observe a small improvement over Figure 33(b). Even for attack messages containing $1K$ legitimate tokens, the SVM and LogitBoost filters generate false negative rates consistently around 7% and 8%, respectively, and the Naïve Bayes filter achieves a false negative rate around



(a) 25 legitimate and 1000 spam features



(b) 25 legitimate and 6000 spam features



(c) 25 legitimate and 9000 spam features

Figure 34: Attack-resistant spam filters. In (a), the filters are built using 25 legitimate and 1000 spam features. In (b), the filters are built using 25 legitimate and 6000 spam features. In (c), the filters are built using 25 legitimate and 9000 spam features. All features are randomly selected.

18%. This resistance to camouflage is better than the retrained filters (without renewed attack) in Figure 29, except for a small range of feature sizes (between 400 and 660) in Figure 29(c).

For completeness, we also include the false positive statistics for this last set of experiments. Figure 34(a) shows the false positives corresponding to Figure 33(a), where the SVM and LogitBoost filters generate around 1.7% false positives, and the Naïve Bayes filter exhibits just over 1%. Figures 34(b) and 34(c) show slight variations in false positive accuracy for the $6K$ and $9K$ trained spam features, between 1.1% and 0.5%. We note that due to the small number of messages involved (0.1% of $5K$ test messages is only 5 messages), changing a single message would explain these variations. In general, the spam producers

are unconcerned about the false positives, and the camouflage attack is only directed at increasing the false negatives. Our experiments corroborate this observation, with very low false positives for every set of experiments described in this chapter.

Finally, it is important to note that the number of retained features (i.e., 25 legitimate and $9K$ spam) suggested by our experimental results may not be appropriate for all corpora or filters. Fundamentally, the success of our proposed spam filter design (and the corresponding heuristics) is significantly more important than the specific feature set sizes that were used to illustrate that success. As spam producers continue to evolve their techniques, we fully expect that victims will be forced to redefine the specifics of our proposed solution; however, the general principles illustrated in this chapter should be applicable for the foreseeable future.

6.7 Summary

The "arms race" between spam producers and victims is created by the victims' attempt to filter out spam and the producers' attempt to circumvent those filters. When learning spam filters such as Naïve Bayes, SVM, and LogitBoost were introduced, they were shown to be successful in distinguishing spam messages. However, spam producers can introduce camouflage attack tokens into spam messages to confuse the learning filters very effectively. The filter retraining approach (using camouflaged messages as spam during the retraining process) may solve the problem for known camouflage attack tokens, but this approach remains ineffective against new, randomized camouflage attack tokens. In Sections 6.3 through 6.5, we described specific rounds of the general spam arms race as well as the camouflage attack/retraining cycle (Round 5 in Section 6.5). These sections include experimental evaluation of the effectiveness of attacks and defenses in each round. For example, baseline filters trained without the knowledge of camouflage quickly succumb to even a small number of camouflage tokens, classifying almost 100% of the camouflaged messages as legitimate (false negatives).

These experiments provided hints for a new method of designing and building spam filters that are resistant to camouflage attacks. This method is primarily based on two

observations. First, we should decrease the filters' sensitivity to camouflage tokens to reduce the effectiveness of camouflaged attack content. Second, we should increase the filters' sensitivity to spam tokens to increase the "distance" between camouflaged spam messages and legitimate messages. This method is implemented with four heuristics: (1) minimize the set of legitimate features L that are used for training to decrease filter sensitivity to camouflage tokens; (2) randomize the content of L to reduce the probability of attack tokens coinciding with L content; (3) limit the size of legitimate email messages to defeat dictionary attacks, and (4) maximize the set of spam features S that are used for training.

Our experiments show that filters (including Naïve Bayes, SVM, and LogitBoost) created using our method are able to resist camouflage attacks effectively (Section 6.6). For example, with 25 legitimate features and $9K$ spam features, we trained learning filters that achieved false negative rates below 10% for attack messages containing $1K$ randomized camouflage tokens. This is a significant and unexpected result since previous work on adversarial classification [42, 104] and experiments on filter retraining (Section 6.5) both suggest that the camouflage attack/retraining cycle will continue for many iterations without an obvious end.

CHAPTER VII

INTEGRATING DIVERSE EMAIL SPAM FILTERING TECHNIQUES

As we have discussed in previous chapters, spam filtering is a commonly accepted technique for dealing with spam, and current spam filters classify messages based primarily on the tokens found in those messages' text. However, this approach has had mixed results. On the one hand, many spam messages have token signatures that facilitate filtering. These signatures typically consist of tokens that are invariant for the many variants automatically generated by spammers. On the other hand, spammers can use various techniques to defeat filters. For example, in Chapter 5, we showed that keyword filters can be defeated using deliberate misspellings, and in Chapters 4 and 6, we found that statistical learning filters can be confused using camouflage (i.e., legitimate content added to spam messages).

To overcome the limitations of token-based filters, we propose a diversified filtering approach that looks at other forms of spam signatures to complement existing text-based techniques. Examples of these other types of signatures include the presence of URLs and the contents of their corresponding Web sites, the file types and contents of email attachments, and header information such as the sender's identity and the email's routed path. In this chapter, we focus our attention on spam messages that contain URLs and provide a novel approach for filtering these messages. The key observation is that most spam messages contain URLs, which are "live" since the spammers would not be able to profit without a functioning link to their site. Thus, by checking the URLs found in a message and verifying a user's interest in the Web sites referenced by those URLs, we are able to add a new dimension to spam filtering.

This chapter has two main contributions. First, we describe three techniques for filtering email messages that contain URLs: URL category whitelists, URL regular expression whitelists, and dynamic classification of Web sites. Second, we describe a prototype implementation that takes advantage of these three techniques to help enhance spam filtering.

Our preliminary results suggest that new dimensions in spam filtering (e.g., using URLs) deserve further exploration.

The remainder of the chapter is structured as follows. Section 7.1 gives an overview of the related work done in this research area. In Section 7.2, we describe our approach, and Section 7.3 discusses the details of our system's implementation. We summarize our findings in Section 7.4.

7.1 Related Work

Spam filtering is currently the standard approach used to stop email spam. However, most of the current approaches and products use content-based filtering. These content-based approaches include whitelisting, blacklisting, keyword-based [35], statistical classification [7, 127], heuristic-based filtering [109, 140], and collaborative filtering [119]. Other classes of filtering approaches include challenge-response [88], MTA/Gateway filtering (Tarproxy [101], greylisting [77], etc.), and micropayments [97, 145].

Some content-based approaches rely exclusively on message headers: automatic and Bayesian whitelisting [93], blacklisting (MAPS, RBL), and others. These techniques have two main disadvantages. Spammers can easily forge message headers, and legitimate domains can easily become blacklisted.

Other content-based approaches rely on message tokens and their corresponding statistics. For example, simple Bayesian Machine Learning approaches, introduced by Duda et al. [47] and originally applied to spam filtering by Sahami et al. [127], use the conditional probability of tokens occurring in spam and legitimate messages to distinguish between these two types of messages. The evaluation by Androutsopoulos et al. [7] showed that these approaches are viable but not without shortcomings. The advantages of these approaches is that they are user-specific and offer low false positive and false negative rates after sufficient training with the current generation of spam. Their disadvantages are that this training process is rather time-consuming, and the resulting training statistics cannot be easily re-used or combined for different users. Additionally, as mentioned in previous chapters, spammers are able to evade these approaches by using common words [104, 151]

and camouflage to augment their spam messages.

Exchange Intelligent Filter [109] and SpamAssassin [140] are two examples of content-based approaches that use heuristics to filter spam. Both tools calculate a score for every message based on manually extracted sets of features (rule bases). The disadvantage of these heuristic-based methods is that they are very ad hoc and require regular updates to ensure accuracy. In fact, these regular updates can often be as complex as the filters themselves [64]. The current version of SpamAssassin attempts to deal with this problem by including a number of plug-ins that support different methods (including Bayesian filtering) to improve the score calculation process. One of these plug-ins is related to our approach: SURBL/SpamCopUri [31]. This plug-in blocks messages by using a blacklist of URLs. The blacklist is created based on the spam submissions received from users. One disadvantage of this method is that it takes time for spam messages to be reported. By the time an update is received, it could already be too late (i.e., other users may have already received the spam messages). Additionally, it is very easy for spammers to change the text of a URL and have it point to the same content (e.g., a redirect).

Our approach is different from SURBL in two ways. First, the definition of what constitutes a spam message in our approach is personalized. Each user has a different set of categories, which correspond to that particular user's interests. Additionally, this information can be combined and shared easily among users. Second, in addition to the text of a URL, our approach also uses the contents of the Web site referenced by that URL. For similar reasons, our approach is also different from the URL module used by Brightmail [135].

7.2 Description

As spammers become more sophisticated, the token signatures (e.g., tokens found in the message body) used by text-based filters to distinguish between spam and legitimate messages will no longer be valid. Thus, we must focus our attention on the characteristics of spam messages that spammers are unable to successfully obscure. A very clear example of such a characteristic is the presence of URLs in spam messages. Spammers rely on these

URLs as a feedback mechanism, and as a result, the URLs must be accessible to the messages' recipients. This required accessibility introduces a new technique for filtering spam messages.

In our new approach, we filter email messages based on the URLs they contain. If the URLs in a particular message point to Web sites that are of interest to a given user, that message is considered legitimate; otherwise, the message is considered spam. We determine a user's interest in a URL (and its corresponding Web site) using three techniques: URL category whitelists, URL regular expression whitelists, and dynamic classification of Web sites. In the following sections, each of these techniques is described in more detail.

7.2.1 URL Category Whitelists

Many search engines (e.g., Google, Yahoo!, LookSmart, etc.) maintain directories that contain category information for Web sites. For example, Google's directory [66] categorizes http://www.google.com as Computers/Internet/Searching/Search Engines/Google. Using these directories, we are able to categorize the URLs a user is interested in, compiling a list of acceptable categories $A_{categories}$. Then, we can use $A_{categories}$ to classify incoming messages as either legitimate or spam based on the URLs they contain. When a new message arrives, the URLs found in that message are categorized. If all of the corresponding categories match categories in $A_{categories}$, the message is classified as legitimate. Otherwise, the message is classified as spam. Unfortunately, not all URLs are listed in the search engines' directories. For the remainder of this chapter, we will use the term *uncategorized* URLs to refer to URLs that are not listed in any of the directories, and we will use the term *categorized* URLs to refer to URLs that are listed in at least one of the directories. In the next section, we describe an additional technique utilized to help handle uncategorized URLs.

7.2.2 URL Regular Expression Whitelists

Using search engine directories to categorize and classify URLs is a novel solution, but it is not always successful. As previously mentioned, uncategorized URLs are not listed in these directories. Additionally, in some cases, users have an interest in Web sites that cannot be expressed easily with categories. For example, a user might want to register

an interest in all Web sites under a given top-level domain (e.g., *.edu, *.mil, etc.). For these special cases, an additional technique is needed to classify the URLs. One possible technique is the construction of a URL regular expression whitelist, which contains a list of acceptable regular expressions A_{regex}. When a new message arrives, the URLs found in that message are compared to the regular expressions in A_{regex}. If all of the URLs match at least one of those regular expressions, the message is classified as legitimate. Otherwise, the system obtains the categories for the URLs that did not match any of the regular expressions and compares those categories to $A_{categories}$ (as explained in the previous section). Unfortunately, this process might still result in uncategorized URLs that do not match any regular expressions in A_{regex}. Thus, in the next section, we explain another technique used to deal with the remaining uncategorized URLs.

7.2.3 Web site Classification

In addition to creating $A_{categories}$ and A_{regex}, our approach also retrieves the contents of the Web sites referenced by the URLs used to create those whitelists. These Web site contents are used to train a learning spam filter (e.g., Naïve Bayes, Support Vector Machines, LogitBoost, etc.), which is used to classify uncategorized URLs that do not match any of the regular expressions in A_{regex}. When a new message arrives containing these uncategorized URLs, the contents of the Web sites referenced by those URLs are retrieved. Then, the learning spam filter is used to classify each of the Web sites. If the filter classifies one of those Web sites as spam, the corresponding message is classified as spam. Otherwise, if all of the Web sites are classified as legitimate, the corresponding message is classified as legitimate.

7.3 Implementation

Our system can be implemented as either server-based or client-based. In the server-based implementation, all users' mail is filtered by a central server, and that server keeps track of each user's profile. In the client-based implementation, each client runs a separate copy of our system. The main advantage of the server-based implementation is the improved performance obtained by maintaining a global cache of URL information (see Section 7.3.1

for a description of this cache). The main advantage of the client-based implementation is the improved privacy protection it provides each user.

For our prototype implementation, we chose a server-based approach. It uses a RedHat Linux v9 machine, which runs an Apache Web server with `mod_ssl`. Our system consists of two parts: a Web-based configuration interface and a mail classifier. The mail classifier is implemented using Perl and procmail, and it also includes a learning spam filter based on POPFile [71]. The system's operation is broken into two phases: training and configuration. These two phases are described in detail in the following sections.

7.3.1 Training

Since each user has unique Web site interests, a separate profile is maintained for every user in the system. Each of these profiles consists of three main parts:

- A list of acceptable categories $A_{categories}$ (as explained above in Section 7.2.1).
- A list of regular expressions A_{regex} for acceptable URLs (as explained above in Section 7.2.2).
- A trained learning spam filter (as explained above in Section 7.2.3).

A user's profile is created during the system's training phase. By default, this profile is generated automatically by the system, but the user also has the option of creating the profile manually. Additionally, our system provides three pre-defined profiles for academic, business, and home users. These profiles can be used without modification; they can serve as a template for users, or they can be disregarded completely. A partial example of the Academic profile is given in Figures 35 and 36.

The automatic profile generation process occurs as follows. First, training URLs are extracted from the user's existing legitimate email messages. Additional URLs can also be obtained from the user's Bookmarks/Favorites list, which is maintained by the user's favorite Web browser. Once the training URLs are obtained, they are categorized using the directories of multiple search engines (e.g., Google, Yahoo!, LookSmart, etc.), and the corresponding categories are stored in $A_{categories}$.

```
Science/Conferences
Science/News
Science/Publications
Computers/Computer_Science/Organizations
Computers/Computer_Science/Research_Institutes
Computers/Computer_Science/Academic_Departments/North_America/United_States/Georgia
Science/Technology/Academia
News/Colleges_and_Universities/
Computers/Internet/Policy
Computers/Internet/Searching/Search_Engines/Google
Computers/Supercomputing
Computers/Systems/
News/By_Subject/Information_Technology
News/By_Subject/Information_Technology/Computers
News/By_Subject/Information_Technology/Internet/Headlines_and_Snippets
...
```

Figure 35: Academic profile: category whitelist.

```
*.edu
*.gov
*.org
*.mil
*.yahoo.com
*.google.com
www.cnn.com
www.nytimes.com
portal.acm.org
ieeexplore.ieee.org
citeseer.ist.psu.edu
www.research.ibm.com
www.research.att.com
www.research.microsoft.com
...
```

Figure 36: Academic profile: regular expression whitelist.

Since it may take several seconds to query the search engines for each URL, our system also maintains a system-wide cache for query results to improve performance. This cache contains category and other information retrieved from search engines, and its entries are periodically expired to ensure the information remains current. Thus, when a user needs to query the search engines, the cache is consulted first. If the necessary information is not found there, the query is forwarded to the search engines.

After the categories are stored in $A_{categories}$ and the system-wide cache, regular expressions are created to match each of the user's unique, uncategorized training URLs. These regular expressions are then stored in A_{regex}. Next, the system retrieves the contents of the Web sites referenced by the URLs that were used to create $A_{categories}$ and A_{regex}. The system also retrieves the contents of the Web sites referenced by the URLs present in the user's existing spam messages. Once the system has these Web sites' contents, those contents are used to train the system's learning spam filter.

An immediate problem that arises when obtaining a Web site's contents is redirection. Spammers can easily use multiple redirects to hide their real Web site. Thus, any attempt to obtain Web site content must handle redirects correctly. In our system, we resolve this

issue by relying on the cached copies of Web sites. which are stored by search engines. When the search engines index Web sites, their crawlers automatically follow the redirects. Thus, the cached copies stored by the search engines are the end-sites rather than the redirecting pages. Our system's learning spam filter uses these cached copies during its training and classification phases.

After the profile generation process is complete, users may edit and verify their $A_{categories}$ and A_{regex} at any time to ensure their interests are properly reflected. They are also able to provide the learning spam filter with additional training data or correct any misclassifications made by the filter.

7.3.2 Classification

Once the training phase is complete, the system uses the user's profile to classify incoming email messages. This classification process works as follows. When a new message arrives, it is scanned for URLs. If the message does not contain any URLs, it is passed to another filtering subsystem, which is beyond the scope of this chapter. Otherwise, the system checks every URL in the message according to the following process.

First, each URL is compared to the regular expressions in A_{regex}. If all of the URLs match at least one of those regular expressions, the message is classified as legitimate. Otherwise, the category information is obtained for the URLs that did not match any of the regular expressions. To improve performance and reduce the load placed on the search engines, the system initially consults the system-wide cache for each URL's category. If the cache does not contain the necessary information, the search engines are queried, and the results are placed in the cache. Once the system has the categories for the URLs, those categories are compared to the categories in $A_{categories}$. For every URL with a category found in $A_{categories}$, a new regular expression is created and added to A_{regex}. The purpose of this new regular expression is to optimize the system's performance when this URL is encountered again, and we refer to this optimization process as *incremental learning*. If all of the URLs have categories found in $A_{categories}$, the message is classified as legitimate. Otherwise, if one of the URLs has a category not found in $A_{categories}$, the message is classified

as spam. However, if some of the URLs are uncategorized, the learning spam filter is used to classify the contents of the Web sites referenced by those URLs. If all of those Web sites are classified as legitimate, the corresponding message is classified as legitimate. Otherwise, if one of the Web sites is classified as spam, the message is classified as spam.

7.4 Summary

In this chapter, we discussed the need for a diversified approach to spam filtering. Concretely, we presented a new method of filtering spam, which focuses on the presence and significance of URLs as a spam signature. URLs are very reliable indicators of spam since they need to be "live" in order for spammers to profit from potential contact with their victims. First, we parse the messages and identify the URLs they contain. Then, we use URL category information already maintained by search engines to check the validity and content classification of those URLs. Based on this information, we are able to determine if the messages are spam.

Our new filtering method complements the current generation of token-based spam filters, which are vulnerable to spammers' manipulation of spam message content. It is also a good example of a diverse and independent technique that can be integrated with other techniques to create a spam filtering system, which is more robust and effective than each of the comprising techniques.

CHAPTER VIII

USING EMAIL SPAM TO IDENTIFY WEB SPAM AUTOMATICALLY

As the Web grew to become the primary means for sharing information and supporting online commerce, the problems associated with *Web spam* also grew. Web spam is defined as Web pages that are created to manipulate search engines and deceive Web users [76, 75]. Just as email spam has negatively impacted the email user experience, the rise of Web spam is threatening to severely degrade the quality of information on the World Wide Web. Web spam is regarded as one of the most important challenges facing search engines and Web users [75, 78], and recent studies suggest that it accounts for a significant portion of all Web content, including 8% of Web pages [54] and 18% of Web sites [76].

Although the problems posed by Web spam have been widely acknowledged, we believe research progress has been limited by the lack of a publicly available Web spam corpus. In previous Web spam research [4, 17, 32, 43, 45, 54, 76, 154], proposed solutions have been evaluated on relatively small Web spam data sets (on the order of hundreds of Web pages). In many cases, these previous researchers had access to large samples of Web data (on the order of millions of pages); however, the onerous task of hand-labeling each Web page made it impossible for them to evaluate even a small fraction of their data. Given the size and dynamic nature of Web content, a manual approach is neither scalable nor sensible. Additionally, none of the previously cited Web spam data sets have been made publicly available so the reproducibility of current Web spam research results is somewhat limited.

Similar to email spam research on the experimental evaluation of spam filters using large spam corpora such as the Enron [95] and SpamArchive [139] corpora, future Web spam research depends on the availability of large collections of Web spam data. Thus, the first contribution of this chapter is a fully automatic Web spam collection technique for extracting large Web spam samples. Our new collection technique is based on the

observation that the URLs found in email spam messages are reliable indicators of Web spam pages. Specifically, we extract the URLs from spam messages, cleanse those URLs of false positives (i.e., URLs for legitimate sites), and collect the corresponding Web pages. Given the dynamic nature of the Web, this collection method is extremely useful because it can be configured to maintain up-to-date Web spam data samples.

The second contribution of this chapter is the Webb Spam Corpus – a large-scale and publicly available Web spam data set that was created using our automated Web spam collection method[1]. This corpus consists of nearly 350,000 Web spam pages, making it more than two orders of magnitude larger than any other previously cited Web spam data set. We describe interesting characteristics of this corpus, and we encourage Web spam and email spam researchers to use it in their work.

The third part of the chapter outlines the usefulness of the Webb Spam Corpus in several application areas. We summarize related research efforts and describe how our Web spam collection technique and corpus could immediately enhance this previous work. Then, we present other interesting application scenarios that we believe could benefit from our approach. One particularly interesting application area is email filtering. Since the Webb Spam Corpus bridges the worlds of email spam and Web spam, we note that it can be used to aid traditional email spam classification algorithms through an analysis of the characteristics of the Web pages referenced by email messages.

The rest of the chapter is organized as follows. Section 8.1 describes our automated technique for obtaining Web spam pages, and it explains how this technique was used to create the Webb Spam Corpus. Section 8.2 provides two sample applications that will immediately benefit from our automated technique and corpus: (1) automatic classification and filtering of Web spam pages and (2) identifying link-based Web spam. Section 8.3 summarizes our findings.

[1]The Webb Spam Corpus can be found at http://www.webbspamcorpus.org/.

8.1 The Webb Spam Corpus

In this section, we describe our method for automatically obtaining Web spam pages, and we present the Webb Spam Corpus – a collection of almost 350,000 Web spam pages that were obtained using our fully automated collection method. First, we explain our general methodology for collecting Web spam. Then, we provide a step-by-step example to clarify the general technique. Finally, we describe the cleansing operations we performed to improve the quality and usefulness of the Webb Spam Corpus.

8.1.1 Obtaining the Corpus

The motivation for our Web spam collection method comes from the observation that email spammers often include URLs in their spam messages. In Chapter 5, we found that at least one URL has appeared in between 85% and 95% of SpamArchive spam messages in all but one month over the course of a three year period. We leverage the presence of those URLs in email spam to aid in the collection of Web spam examples.

8.1.1.1 General Methodology

As previously mentioned, our Web spam collection technique relies on the URLs found in email spam messages. We obtained our email spam messages from the SpamArchive corpora[2]. Between November 2002 and January 2006, SpamArchive collected and published close to two thousand archives (each stored as a gzipped mbox folder), totaling more than 1.4 million spam messages. We used all of these messages to help obtain the Webb Spam Corpus.

First, we downloaded all of the SpamArchive archives that were published up until January 6, 2006, and we gunzipped these archives to obtain their corresponding mbox folders. Then, we parsed the messages in each mbox folder to obtain a list of the unique URLs that were present in the "Subject" headers and bodies of those messages. We extracted URLs from arbitrary text using Perl's URI::Find::Schemeless module, and we used Perl's Html::LinkExtor module to extract URLs from HTML. In total, we extracted almost 1.2

[2]SpamArchive's spam corpora can be found at ftp://spamarchive.org/pub/archives/.

Table 18: Number of redirects returned by intended URLs.

Number of Redirects	Number of Intended URLs
0	204,077
1	92,952
2	37,789
3	7,230
4	4,358
5	1,120
6	585
7	322
8	117
9	55
10	61
11	193
12	13
13	6

million unique URLs, which we will refer to as the *intended URLs*. Once we had this list of intended URLs, we wrote a custom crawler (based on Perl's LWP::Parallel::UserAgent module) to obtain their corresponding Web pages.

The crawler attempted to access each of the intended URLs; however, many of the URLs returned HTTP-level redirects (i.e., a 3xx HTTP status code). The crawler followed all of these redirects until it finally accessed a URL that did not return a redirect. We refer to this final URL in the redirect chain as an *actual URL*, and it is important to note that all of the intended URLs that were successfully accessed without returning a redirect (i.e., they returned 2xx HTTP status codes) are also considered actual URLs. To illustrate the amount of redirection that occurred, Table 18 shows the number of intended URLs that returned between 0 and 13 redirects. At the top of the table, we observe that the majority (204,077) of the intended URLs were also actual URLs, and at the bottom of the table, we observe that 6 intended URLs forced the crawler to follow 13 redirects before it accessed the actual URL.

Our crawler obtained two types of information for every successfully accessed URL (including those that returned a redirect): the HTML content of the page identified by

the URL and the HTTP session information associated with the page request transaction. The HTTP session information contains a number of headers, but the exact content varies from page to page. The most common headers include the HTTP status code, "Server" (the version of the Web server that served the page), "Content-Type", and "Client-Peer" (the IP address of the machine that served the page). In addition to the standard headers obtained by the crawler, we also added a header for a page's URL using the following format:

URL: <*URL of the page*>

We stored each of these HTTP session headers in a file by using HTML's commenting mechanism (i.e., <!−− HEADER −−>) to ensure each file is a valid HTML document (and parseable by an HTML parser). For example, in the actual corpus files, "Content-Type: text/html" is stored as "<!−− Content-Type: text/html −−>".

For each of the actual URLs, the corresponding HTML content and session information are both stored in a single file that abides by the following naming convention:

<*md5 hash of the page's HTML content*>_*m*,

where m is an integer value used to uniquely identify files that share the same md5 hash value for their HTML content. For example, MD5_0 contains the first page with MD5 as the md5 hash value for its HTML content, MD5_1 contains the second page with this md5 value, and so on.

For each of the intended URLs that has an associated redirect chain, the HTML content (if any exists) and session information for each link in the redirect chain are both stored in a single file that abides by the following naming convention:

<*md5 hash of the page's HTML content*>_*m*_redirect_*n*,

where m is the same as above, and n is an integer used to uniquely identify each link in a given redirect chain. For example, MD5_0_redirect_0 contains the original response that was obtained from the intended URL, and MD5_0_redirect_1 contains the response that was obtained from the next link in the chain. This pattern continues for as many links as there were in the redirect chain. As explained above, MD5_0 contains the page that

```
From nobody Wed May 28 18:53:38 2003
Return-Path: <bounce-53821024-5237@mail22.recessionspecials.com>
Received: from mail22.recessionspecials.com ([65.61.148.12])
    by (InterMail vM.5.01.05.17 201-253-122-126-117-20021021) with SMTP id
    for  Sat, 22 Mar 2003 03:23:44 -0700
Content-Disposition: inline
Content-Transfer-Encoding: 7bit
Content-Type: text/plain; boundary="_----------=_364530249436920041706 6"
MIME-Version: 1.0
X-Mailer: MIME::Lite 2.117  (F2.6; B2.12; Q2.03)
Date: Sat, 22 Mar 2003 10:23:43 UT
Subject: You've Won!
X-List-Unsubscribe: <unsub-53821024-5237@recessionspecials.com>
From: "Vicki" <returns-lztfkoskhyzktw@recessionspecials.com>
Reply-To: "Vicki" <returns-lztfkoskhyzktw@recessionspecials.com>
Return-Path: <bounce-53821024-5237@recessionspecials.com>
To: submit@spamarchive.org

You've Won!

Click to see what it is:
http://click.recessionspecials.com/sp/t.pl?id=92408:57561182
------------------------------------------------

Remove yourself from this recurring list by sending a blank email to
mailto:unsub-53821024-5237@recessionspecials.com
```

Figure 37: Example email spam message obtained from SpamArchive.

corresponds to the end of the chain (i.e., the response obtained from the actual URL). Also, by extracting the "URL" value from each file's session information, it is possible to traverse the path of links that lead from the intended URL to the actual URL.

We used the md5 hash of each page's HTML content as the primary file name information to facilitate efficient duplicate page detection within the corpus. However, it is important to note that we have not actually removed any duplicate pages from the corpus. Since each of the intended URLs was unique, the duplicate pages in the corpus imply multiple entrances (or gateways) to the same page. This situation is very similar to Web spamming techniques such as link exchanges and link farms, and in Section 8.2.2, we will investigate this observation further. We believe these types of observations are extremely interesting and quite useful for investigating the techniques that are being used by Web spammers. Thus, we have tried to keep perturbations of the corpus to a minimum. Unfortunately, some corpus cleansing operations were unavoidable, and we describe those operations in Section 8.1.2.

8.1.1.2 Illustrative Example

To help clarify our general methodology, we provide a step-by-step explanation (with examples) of our automatic Web spam collection technique. We begin this description with an example of an email spam message that we obtained from SpamArchive. Figure 37 shows

```
<!-- URL: http://click.recessionspecials.com/sp/t.pl?id=92408:57561182 -->
<!-- HTTP/1.1 302 Found -->
<!-- Connection: close -->
<!-- Date: Fri, 23 Dec 2005 18:11:43 GMT -->
<!-- Location: / -->
<!-- Server: Apache/2.0 -->
<!-- Content-Length: 0 -->
<!-- Content-Type: text/html; charset=UTF-8 -->
<!-- X-Powered-By: PHP/5.0.5 -->
```

Figure 38: Example HTTP session information obtained from an HTTP redirect.

the headers and body of this spam message.

Upon obtaining this message, we parsed its "Subject" header and message body text to obtain a list of intended URLs. In this example, two intended URLs were found:

`http://click.recessionspecials.com/sp/t.pl?id=92408:57561182`

and

`mailto:unsub-53821024-5237@recessionspecials.com.`

We rejected the second URL because we only retained URLs with "http" or "https" as their scheme. It is important to note that many URLs were schemeless, and we retained all of those URLs.

After we parsed `http://click.recessionspecials.com/sp/t.pl?id=92408:57561182` as an intended URL, we used our crawler to obtain its corresponding Web page. However, this URL returned a redirect that directed the crawler to `http://click.recessionspecials.com/`. Figure 38 shows the HTTP session information (it did not have any HTML content) associated with this redirect. As described above in Section 8.1.1.1, this session information provides valuable information about the HTTP page request transaction. For example, the figure shows the HTTP status code (302), the type of Web server that processed the request (Apache/2.0), and the next URL in the redirect chain (`http://click.recessionspecials.com/`).

Upon receiving the redirect, our crawler attempted to obtain the Web page corresponding to the next URL in the redirect chain. This next URL did not return a redirect (i.e., it is an actual URL) so the crawler successfully obtained the page. Figure 39 shows the HTTP session information and HTML content associated with this Web spam page.

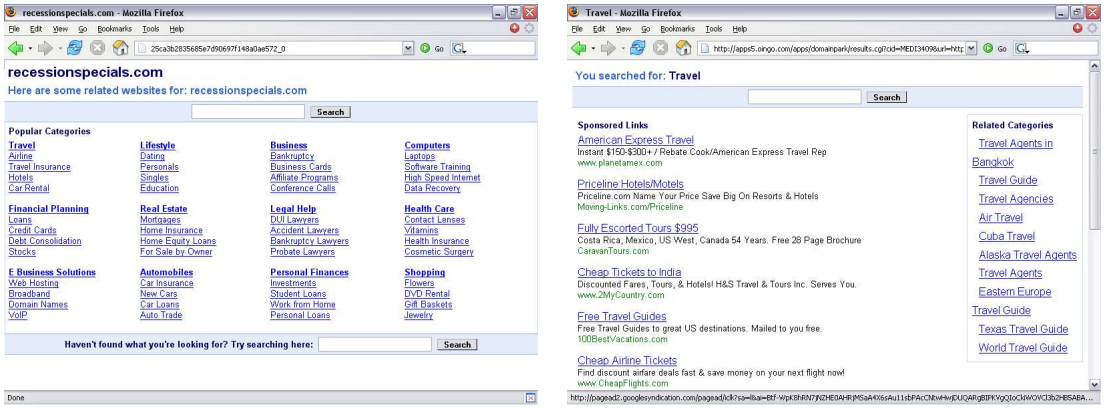
The md5 hash value for this page's HTML content is 25ca3b2835685e7d90697f148a0ae572. Thus, we used this md5 value to name all of the corpus files associated with this page. The data shown in Figure 38 is stored in a file named 25ca3b2835685e7d90697f148a0ae572_0_redirect_0. Similarly, the information shown in Figure 39 is stored in a file named 25ca3b2835685e7d90697f148a0ae572_0.

```
<!-- URL: http://click.recessionspecials.com/ -->
<!-- HTTP/1.1 200 OK -->
<!-- Connection: close -->
<!-- Date: Fri, 23 Dec 2005 18:12:19 GMT -->
<!-- Server: Apache/2.0 -->
<!-- Content-Length: 732 -->
<!-- Content-Type: text/html; charset=UTF-8 -->
<!-- Client-Peer: 64.40.102.44:80 -->
<!-- Link: <http://static.hitfarm.com/template/qing/images/qing.ico>;
/="/"; rel="shortcut icon"; type="image/x-icon" -->
<!-- P3P: CP="NOI COR NID ADMa DEVa PSAa PSDa STP NAV DEM STA PRE" -->
<!-- Set-Cookie: source=1; expires=Fri, 23 Dec 2005 20:12:19 GMT -->
<!-- Title: recessionspecials.com -->
<!-- X-Powered-By: PHP/5.0.5 -->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="en">
<head>
<title>recessionspecials.com</title>
<link rel="shortcut icon" href="http://static.hitfarm.com/template/qing/
images/qing.ico" type="image/x-icon" />
</head>
<frameset cols="1,*" border="0" frameborder="0">
  <frame name="hftop" src="/top.php" scrolling="no" frameborder="0"
    marginwidth="0" marginheight="0" noresize="noresize" />
  <frame name="hfasi" src="http://apps5.oingo.com/apps/domainpark/
    domainpark.cgi?cid=MEDI3409&s=recessionspecials.com&ip=130.207.5.18"
    scrolling="auto" frameborder="0" marginwidth="0" marginheight="0"
    noresize="noresize" />
<noframes>
<body>
<p>This page requires frames</p>
</body>
</noframes>
</frameset>
</html>
```

Figure 39: Example HTTP session information and content for a Web spam page.

To investigate this Web spam page further, we rendered its HTML content in a popular Web browser (Firefox). Figure 40(a) shows the browser-rendered view of the HTML file shown in Figure 39. This figure clearly shows that the page is an example of a fake directory (also known as a directory clone [75]) – a seemingly legitimate page that contains a vast number of outgoing links to other pages, grouped into categories. Legitimate directories (e.g., the DMOZ Open Directory) attempt to provide users with an unbiased, categorized view of the Web. Fake directories also provide a categorization of the Web, but it is biased by the motivations of the Web spammer. Figure 40(b) shows the browser-rendered view of the page that is returned after a user clicks on the "Travel" link (located at the top-left of the original page). This page is filled with travel-related links; however, all of the links are tied to Google's AdSense program. Thus, the Web spammer receives a monetary reward every time a user clicks on one of these links. Also, as is often the case with fake directories, some of these links point to pages that are controlled by the Web spammer. Thus, this spamming technique also generates additional traffic for the spammer's other pages. This example illustrates one of the many interesting observations that can be made with the aid of our technique and corpus. In Section 8.2, we will investigate other areas where our work



(a) (b)

Figure 40: Examples of browser-rendered Web spam pages.

can be applied.

8.1.2 Cleansing the Corpus

In this section, we describe the cleansing operations we performed to make the Webb Spam Corpus as useful as possible. These cleansing operations fall into two categories: "Content-Type" purification and false positive removal. During the crawling process, we obtained 407,987 files that had a number of different values in their "Content-Type" session headers (e.g., application, audio, image, video, etc.). Since we were only interested in maintaining HTML Web pages in the Webb Spam Corpus, we removed all of the files with non-textual "Content-Type" headers (25,065 content files and their corresponding 33,312 redirect files).

After we removed the non-textual files from the corpus, we began analyzing the remaining 382,922 Web pages. During this analysis, we found a number of duplicates. Specifically, of the 382,922 spam Web pages, 101,453 were duplicates, and 281,469 were unique. Duplicate pages exist in the corpus because a number of intended URLs directed our crawler to the same actual URL. Thus, each of the duplicate pages has a unique redirect chain associated with it. Figure 41 shows the distribution of intended URLs per actual URL (i.e., the number of intended URLs that point to the same actual URL) that we found during our initial analysis. The x-axis shows how many intended URLs mapped to a single actual URL, and the y-axis shows how many of these actual URLs there were. A point at position

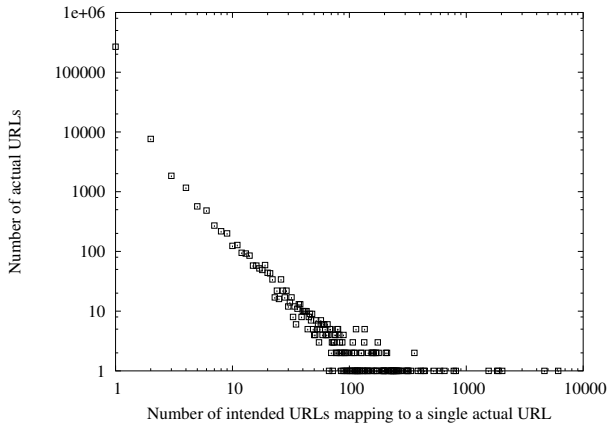


Figure 41: Distribution of the number of intended URLs that point to the same actual URL.

(x, y) denotes the existence of y actual URLs such that each actual URL was pointed to by x intended URLs.

From the figure, we see that 267,533 actual URLs were uniquely pointed to by one intended URL (the point at the top-left of the figure), and 7,628 actual URLs were pointed to by two intended URLs. On the opposite end of the spectrum, one particular actual URL was pointed to by 6,108 intended URLs (the point at the bottom-right of the figure). To further investigate this point and others at the bottom-right of the figure, Table 19 provides the 10 actual URLs that were pointed to by the most intended URLs. In this table, each actual URL is provided along with the number of intended URLs that pointed to it.

This table clearly shows a number of specific Web spam examples; however, it also shows two unexpected URLs:

`http://www.yahoo.com`

and

`http://www.msn.com`.

Most people would agree that neither of these URLs are spam so their presence in the corpus was somewhat confusing. To explain this phenomenon, we investigated the intended URLs that pointed to these two actual URLs and found that they were broken redirects. The following two URLs are examples of the types of broken redirects we found:

`http://drs.yahoo.com/quips/*http://coolguy.biz/sm/chair.php`

Table 19: Ten actual URLs that were pointed to by the most intended URLs.

Actual URL	Count
http://blueroc ketonline.TechBuyer.com/ landerjsp?referrer=&domain=bluerocketonline.com&cm_mmc=	6,108
http://www.yahoo.com	4,663
http://www.msn.com	2,028
http://yoursmartrewards.com/ rd_p?p=99302&c=9479-visa300_emc_d22&a=CD579	1,880
http://click.recessionspecials.com/	1,836
http://migada.com/main_index.html	1,553
http://gmncc.com/main_index.html	813
http://click.beyondspecials.com/	783
http://web.yearendsaver.com/	597
http://mail02a.emailcourrier.com/	526

and

`http://g.msn.com/8HMBEN/?http://www.all-net-offers.com/jf.`

Email spammers used these URLs in their messages to deceive users. At first glance, the URLs appear to be legitimate Yahoo! and MSN URLs, and as a result, unsuspecting users might click on them. Unfortunately, when our crawler tried to access these URLs, the targets of the redirection were no longer available. Consequently, the legitimate Yahoo! and MSN URLs were inserted into our corpus multiple times.

Upon discovering these anomalous pages, we devised a few heuristics to identify additional false positives (i.e., legitimate pages that were mistakenly included in the corpus). Specifically, we used Alexa's Top 500 list [3] and SiteAdvisor's rating system [137] to compile a list of potential false positives. Then, we manually inspected this list and identified the pages that were clearly false positives. Finally, we removed the false positives that we identified. After we applied this cleansing process to the corpus, we found and removed 34,044 legitimate pages as well as their 44,141 corresponding redirect files, leaving the Webb Spam Corpus with a final total of 348,878 Web spam pages and 223,414 redirect files.

The presence of false positives in our original collection raises an interesting research challenge. Since our approach relies upon the URLs found in email spam messages, we

believe it is potentially vulnerable to a new breed of attacks called *legitimate URL attacks*. In these attacks, spammers intentionally include legitimate URLs (e.g., `www.yahoo.com`) in their email spam messages to introduce false positives into our Web spam collection process. This potential attack has not affected the Webb Spam Corpus, but it could impact future efforts to collect Web spam using our methodology as well as other automated spam collection efforts (e.g., SpamArchive). Thus, we present this potential threat as an important direction for future research.

8.2 Sample Applications

In this section, we focus on sample applications that will greatly benefit from our corpus and Web spam collection methodology. Our work is broadly applicable to Web spam research, but we focus our attention on two specific applications: (1) automatic classification and filtering of Web spam pages and (2) identifying link-based Web spam.

8.2.1 Automatic Classification and Filtering of Web Spam Pages

As long as the Web has been in existence, researchers have been developing techniques to automatically categorize its content. Originally, these categorization efforts were concerned with automatically producing Web directories with minimal human interaction. However, due to the emergence of Web spam, researchers have begun focusing their efforts on automatically classifying Web spam pages to separate them from legitimate Web pages. The evolution of these classification efforts is very similar to the evolution of early email spam classification research. Similar to the limitations faced by email spam researchers before the introduction of the Ling-spam [6], PU123A [7, 9], and Enron [95] corpora, current Web spam classification research is limited by the lack of a publicly available corpus. In the next section, we describe several previous Web spam classification research efforts. Then, we discuss how our corpus and Web spam collection process will contribute to this previous work. Finally, we describe how our work will further future research in this area.

8.2.1.1 Summary of Related Work

Chandrinos et al. [32] were among the first to investigate automatic filtering of Web spam. Specifically, they presented a method for automatically identifying pornographic content on the Web. First, they trained a Naïve Bayesian classifier to distinguish between obscene and non-obscene Web pages. To train the classifier, they used the textual contents of the page as well as two image attributes: whether or not the page contained at least one suspicious image and whether or not the page contained at least one non-suspicious image. Then, they used this classifier to determine whether or not a given page was obscene. Using a collection of 849 pages (315 legitimate pages and 534 pornographic pages), their classifier was able to obtain 100% obscene precision and 97.5% obscene recall (with zero obscene false positives).

Amitay et al. [4] presented a unique approach for categorizing Web sites. Instead of focusing on the content of the sites, they only utilized connectivity and structural data to categorize sites into eight pre-defined functionality categories (e.g., corporate sites, search engines, portals, etc.). First, they used their connectivity and structural data to identify 16 distinguishing features. Then, using these features, they utilized two automatic learning techniques to categorize the sites: a decision-rule classifier and a Bayesian classifier. Using a data set of 202 manually tagged Web sites, their decision-rule classifier achieved an average expected error of 45.5%, and their Bayesian classifier achieved an average expected error of 41.0%. Also, although it was not the focus of their research, they proposed using their technique to classify Web spam pages.

Fetterly et al. [54] and Drost and Scheffer [45] both used statistical techniques to identify Web spam pages. Fetterly et al. statistically analyzed two large data sets of Web pages (DS1 and DS2) using properties such as linkage structure, page content, and page evolution. They found that many of the outliers in the statistical distribution of these properties were Web spam, and as a result, they advocated the use of outlier detection for identifying similar pages. Drost and Scheffer trained a Support Vector Machine (SVM) classifier to accurately distinguish between guestbook spam, link farms and link exchange sites, and legitimate sites. In their evaluations, they used 854 DMOZ Open Directory pages as their legitimate sample and 431 manually identified Web spam pages (251 examples of guestbook spam

and 180 link farm and link exchange pages). The results of their evaluations showed that the SVM classifier can effectively separate spam and legitimate Web pages, consistently obtaining area under the ROC curve (AUC) values greater than 90%.

8.2.1.2 Benefits of the Webb Spam Corpus

All of this previous research is novel; however, it suffers from two main limitations. First, all of the data sets used in these evaluations are far too small to be considered representative samples of Web spam. These data sets are small because Web spam researchers have been forced to manually identify and tag Web spam examples. This manual labeling process is extremely time-consuming, and as a result, it has forced previous researchers to apply their techniques on limited samples of their Web data. Second, none of the previously cited Web spam data sets have been released into the public domain. Thus, other researchers are currently unable to verify the validity of the claims made by this previous research. In their paper, Drost and Scheffer claim to have a publicly available spam data set; however, the paper does not provide a link, and we have been unable to locate it online.

Our Web spam collection technique and corresponding corpus help solve both of the limitations found in previous research. The Webb Spam Corpus is a very large sample of Web spam (over two orders of magnitude larger than previously cited Web spam data sets). Also, our automated Web spam collection technique allows researchers to quickly and easily obtain even more examples. The main challenge with any automated Web spam classification technique is accurate labeling (as shown by the limited Web spam sample sizes of previous research), and although our approach does not completely eliminate this problem, it does minimize the manual effort required. Researchers simply need to identify a few false positives as opposed to the arduous task of manually searching for a sufficiently large collection of Web spam pages. Additionally, the Webb Spam Corpus is publicly available so researchers can easily publish reproducible results.

8.2.1.3 New Research Opportunities

In addition to the benefits our approach and corpus offer previous research efforts, we believe this work opens the door for a number of new research opportunities. First, automatic Web

spam classification could greatly benefit Web filtering efforts, similar to the way email spam classification has improved email filtering. Specifically, our work could be used to provide more effective parental controls on the Web. The Webb Spam Corpus contains a number of porn-related pages as well as additional content that is not suitable for children. This content provides valuable insight into the characteristics of Web spam pages and allows researchers to build more effective Web content filters.

In addition to its contributions to Web filtering, the Webb Spam Corpus also provides a unique approach to email spam filtering. In Chapter 7, we showed how Web content can be used to improve the effectiveness of email spam filtering techniques. Specifically, we leveraged the Web content that corresponds to the URLs found in email messages to enhance email classification decisions. An email message that points to legitimate Web pages is more likely to be legitimate than an email message that points to suspicious Web pages. By augmenting traditional text-based spam filters with contextual information such as the spamicity of the URLs found within an email message, we can create more sophisticated classification systems. Thus, we can utilize the link between email and the Web to help eliminate spam in both domains.

8.2.2 Identifying Link-Based Web Spam

One of the most prominent examples of Web spam is the targeted manipulation of search engines to increase the visibility of certain Web spam pages. Since the vast majority of Web users use search engines to access the Web [33, 122], spammers can artificially increase the value of a spam page by effectively deceiving a search engine's core ranking algorithms. Most modern search engines employ link-based ranking algorithms (e.g., Google's PageRank) that evaluate the quality of a Web page based on the number and quality of the other Web pages that point to it. These algorithms rely on a fundamental assumption that a link from one page to another is an authentic conferral of authority by the pointing page to the target page.

Link-based Web spam directly attacks the credibility of such link-based ranking algorithms by inflating the perceived quality of the spammer's target page. The simplest example of link-based Web spam is called a *link exchange* – a scenario in which two or more Web spammers collude to link to each other's pages. A more sophisticated example is the construction of large *link farms* of spammer-controlled Web pages that exist solely to link to a spammer's target page. These link farms make it appear to the link-based ranking algorithms that the target page is receiving many votes of confidence, and as a result, the target page receives an undeserved boost in its ranking. In contrast to the brute force approach of a link farm (where there are a large number of links from low-quality, spammer-controlled pages), spammers also engage in techniques to acquire links from higher-quality Web pages. For example, a Web spammer can create a seemingly legitimate Web site (called a *honey pot*) to attract links from unsuspecting legitimate Web sites. The honey pot can then pass along its accumulated quality to the target spam page. Spammers can also *hijack* links from legitimate Web sites by inserting links into Weblog comments, wikis, Web-based message boards, as well as submitting spam links to legitimate Web directories. From the link-based ranking algorithm's perspective, these hijacked legitimate pages are endorsing the spam page, and as a result, the spam page receives an undeserved ranking boost. Of course, Web spammers may choose to combine these basic link-based Web spam patterns in a number of ways to construct more complicated and less easily detected linking arrangements. For a more detailed discussion of these spamming techniques, please consult [75].

The majority of previous link-based Web spam research has focused on its identification and removal; however, we believe this research has been limited by the absence of a publicly available Web spam corpus. In the next section, we summarize several of these previous efforts and explain their limitations. Then, we explain how our Web spam collection technique and corpus will enhance this previous work and help facilitate future work on the identification of link-based Web spam.

8.2.2.1 Summary of Related Work

Davison [43] was the first to investigate link-based Web spam, and he studied the identification of *nepotistic links* – "links between pages that are present for reasons other than merit." Specifically, he created decision trees to determine whether or not a given link was nepotistic. His experiments relied on two data sets – 1,536 links that were arbitrarily selected and 750 links that were sampled from a DiscoWeb search engine crawl. This work was extremely valuable because it was the first to use automated learning methods to identify link-based Web spam. However, it also identified two corpora-related problems with Web spam research. First, as the author admits, the two data sets were too small to be considered representative samples. Second, the results obtained with each data set were noticeably different, highlighting the need for a publicly available Web spam corpus to help benchmark research results.

These corpora-related limitations are also evident in other previous research. For example, the TrustRank algorithm, proposed by Gyöngyi et al. [76], uses a variation of the traditional PageRank algorithm to propagate trust from a seed set of pre-trusted Web pages to the pages that are pointed to by those pre-trusted pages. Intuitively, pages that have many incoming-links from trusted pages will also be trusted. As long as spam pages are relatively distant (in terms of link distance) from trusted pages, the algorithm can yield more spam resilient rankings than PageRank. Unfortunately, although the TrustRank experimental validation had access to a data set of 31 million Web sites that were collected by AltaVista, the actual experiments only used an evaluation sample of 748 manually identified pages (of which, 135 were labeled as spam). The limited size of this evaluation size is another illustration of the difficulty involved with manually identifying Web spam examples.

Wu and Davison [154] proposed a technique that propagates distrust to bad pages. First, their algorithm searches for pages with common nodes in their incoming and outgoing links sets. If the number of common nodes for a given page is above a given threshold (3, in their experiments), that page is marked as bad and placed in a seed set. Then, every page that points to more than a threshold number (3, in their experiments) of pages in the seed set is also marked as bad. Finally, every link involving one of the bad pages is considered spam.

In their experiments, Wu and Davison had access to a 20 million page data set that they received from the search.ch search engine; however, their evaluation sample only contained 732 spam sites (due to the manual labeling problem).

Similarly, Benczur et al. [17] proposed the SpamRank algorithm to identify pages with a large amount of undeserved PageRank (i.e., PageRank that was derived from spam pages). First, they identified the major PageRank contributing pages (the "supporters") for every page in their data set. They then penalized pages that had statistically anomalous supporters (in terms of the distribution of their PageRank scores). By incorporating these penalties into the revised PageRank calculation, pages with a large amount of undeserved PageRank were identified and given much lower PageRank values (and correspondingly lower rankings). The authors report experimental results over an evaluation sample containing 910 pages (of which 16.5% were spam) that were taken from a 31 million page data set.

8.2.2.2 Benefits of the Webb Spam Corpus

Given the interesting research results so far, we believe that rich, new opportunities exist for cross-validating previous results, enhancing the previously proposed link-based Web spam algorithms, and developing more refined algorithms for identifying link-based Web spam. Our new method for obtaining Web spam examples (and the Webb Spam Corpus itself) immediately benefits this previous research because it greatly increases the coverage of Web spam pages. Our corpus already contains over two orders of magnitude more Web spam examples than previous data sets (almost 350,000 pages versus less than 2,000 in each cited case), and unlike those previous data sets, our corpus is publicly available.

With our corpus, researchers can easily benchmark their techniques using a single, publicly available corpus – a luxury currently missing from Web spam research. Additionally, our collection methodology gives researchers a simple technique for automatically obtaining new Web spam examples in the future, a particularly important feature for maintaining the freshness of spam samples in the face of a dynamic and evolving group of spam adversaries. This automatic technique should greatly reduce the workload of Web spam researchers, minimizing the manual Web spam tagging process and allowing them to focus their efforts

on developing new solutions.

8.2.2.3 New Research Opportunities

In addition to providing a standardized corpus for evaluating link-based Web spam identification algorithms, we believe that a careful study of the linking features of the Webb Spam Corpus may yield new insights into how spammers construct complex linking arrangements and provide new avenues for developing more robust link-based Web spam identification algorithms.

As a first step towards developing new algorithms, we propose the following hypothesis: many of the Web pages corresponding to URLs found in spam messages are also target pages in link-based Web spam. This hypothesis is driven by the observation that email spammers and Web spammers share the same motivations. In email spam, spammers want to promote certain pages so that they receive traffic and ultimately monetary rewards (or private information, in the case of phishers). Similarly, in Web spam, spammers want to promote certain pages for the exact same reasons.

To help investigate this hypothesis, we constructed a host-based connectivity graph to determine the interconnectivity within the Webb Spam Corpus. Initially, we treated each host that appears in the Webb Spam Corpus as a node in a Web graph. Then, we parsed each page in the corpus to obtain the URLs found in its HTML content, only retaining the URLs that pointed to another page within the corpus. Each of these URLs represents a link from one page in the corpus to another. After we had all of these page-level links, we converted them to host-level links (based on the host names in each URL) to make the number of nodes in the Web graph more manageable. Finally, we constructed a host-based connectivity graph.

The host-based connectivity graph contains 70,230 unique hosts and 137,039 unique links (not including self-links). Thus, by simply investigating the hosts within the Webb Spam Corpus, we have already identified a great deal of interlinkage. We have been forced to omit the complete host-based connectivity graph because it looks like a giant circle of ink (due to the vast number of nodes and interconnections). Instead, we have provided an

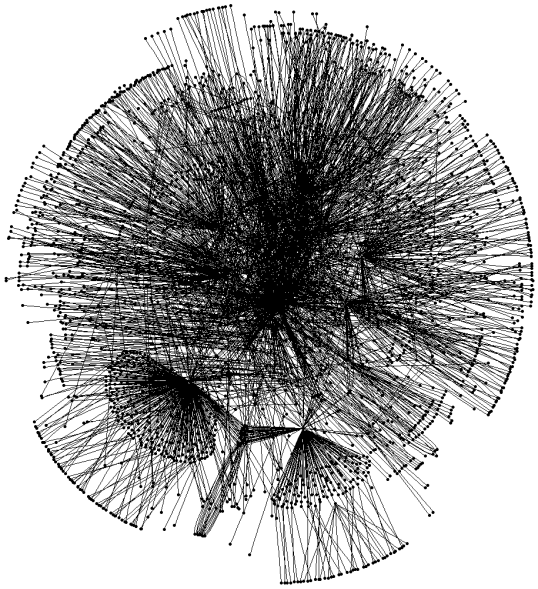


Figure 42: Host-based connectivity graph.

extremely condensed version of the connectivity graph in Figure 42.

The graph shown in Figure 42 was constructed using the 15 hosts with the most outgoing links (i.e., they linked to the largest number of hosts in the corpus). The figure shows each of the hosts along with their 1-hop (hosts they directly link to) and 2-hop (hosts their direct neighbors link to) neighbors. Even this simple graph, which contains 948 unique hosts and 3,330 unique links, clearly illustrates the interconnectivity of the Web spam hosts within the corpus. We believe this interconnectivity provides preliminary support for our hypothesis, and as a result, we intend to thoroughly investigate this topic in future research.

8.3 Summary

As the problems posed by Web spam continue to grow in severity, it is imperative that the research community follow the best practices that have already been established in similar domains (e.g., email spam research). Of these best practices, one of the most important is the use of large, publicly available corpora. In this chapter, we have taken the initial step towards applying these best practices to the Web spam domain. We have provided a novel method for automatically obtaining Web spam pages, and we have also presented the Webb Spam Corpus – a publicly available corpus of almost 350,000 Web spam pages that were

obtained using our automated method.

The Webb Spam Corpus is the first public data set of its kind, and it is more than two orders of magnitude larger than previously cited Web spam data sets. In all research domains, the lack of publicly available corpora severely impedes research progress; thus, by presenting our approach to Web spam collection and the Webb Spam Corpus, we hope to fuel significant research efforts.

CHAPTER IX

CHARACTERIZING WEB SPAM USING CONTENT AND HTTP SESSION ANALYSIS

Web spam has grown to a significant percentage of all Web pages (between 13.8% and 22.1% of all Web pages [27, 115]), threatening the dependability and usefulness of Web-based information in a manner similar to how email spam has affected email. Unfortunately, previous research on the nature of Web spam [27, 54, 115, 149, 150, 153] has suffered from the difficulties associated with manually classifying and separating Web spam pages from legitimate pages. As a result, these previous studies have been limited to a few thousand Web spam pages, which is insufficient for an effective content analysis (as customarily performed in email spam research).

In this chapter, we provide the first large-scale experimental study of Web spam pages by applying content and HTTP session analysis techniques to the Webb Spam Corpus – a collection of almost 350,000 Web spam examples that is two orders of magnitude larger than the collections used in previous evaluations. Our main hypothesis in this study is that Web spam pages are fundamentally different from "normal" Web pages. To evaluate this hypothesis, we characterize the content and HTTP session properties of Web spam pages using a variety of methods. The Web spam content analysis is composed of two parts. The first part quantifies the amount of duplication present among Web spam pages. Previous studies [24, 53, 55] have shown that only about two thirds of all Web pages are unique; thus, we expected to find a similar degree of duplication among our Web spam pages. To evaluate duplication in the corpus, we constructed clusters (equivalence classes) of duplicate or near-duplicate pages. Based on the sizes of these equivalence classes, we discovered that duplication is twice as prevalent among Web spam pages (i.e., only about one third of the pages are unique).

The second part of the content analysis focuses on a categorization of Web spam pages.

Specifically, we identify five important categories of Web spam: **Ad Farms**, **Parked Domains**, **Advertisements**, **Pornography**, and **Redirection**. The **Ad Farms** and **Parked Domains** categories consist of pages that are comprised exclusively of advertising links. These pages exist solely to generate traffic for other sites and money for Web spammers (through pay-per-click advertising programs). The **Advertisements** category contains pages that advertise specific products and services, and the pages in the **Pornography** category are pornographic in nature. The **Redirection** category consists of pages that employ various redirection techniques. Within the **Redirection** category, we identify seven redirection techniques (HTTP-level redirects, 3 HTML-based redirects, and 3 JavaScript-based redirects), and we find that 43.9% of Web spam pages use some form of HTML or JavaScript redirection.

The third component of our research is an evaluation of the HTTP session information associated with Web spam. First, we examine the IP addresses that hosted our Web spam pages and find that 84% of the Web spam pages were hosted on the 63.* – 69.* and 204.* – 216.* IP address ranges. Then, we evaluate the most commonly used HTTP session headers and values. As a result of this evaluation, we find that many Web spam pages have similar values for numerous headers. For example, we find that 94.2% of the Web spam pages with a "Server" header were hosted by Apache (63.9%) or Microsoft IIS (30.3%). These results are particularly interesting because they suggest that HTTP session information might be extremely valuable for automatically distinguishing between Web spam pages and normal pages.

The rest of the chapter is organized as follows. Section 9.1 describes our Web spam corpus and summarizes its collection methodology. In Section 9.2, we report the results of a content analysis of Web spam, which consists of two parts. The first part evaluates the amount of duplication that appears in Web spam. The second part identifies concrete Web spam categories and provides an extensive description of the redirection techniques being used by Web spammers. In Section 9.3, we report the results of an analysis of Web spam HTTP session information, which identifies the most common hosting IP addresses and HTTP header values associated with Web spam. Section 9.4 summarizes related work,

and Section 9.5 summarizes our results.

9.1 Corpus Summary

In the previous chapter, we presented an automatic technique for obtaining Web spam examples that leverages the presence of URLs in email spam messages. Specifically, we extracted almost 1.2 million unique URLs from more than 1.4 million email spam messages. Then, we built a crawler to obtain the Web pages that corresponded to those URLs. Our crawler attempted to access each of the URLs; however, many of the URLs returned HTTP redirects (i.e., 3xx HTTP status codes). The crawler followed all of these redirects until it finally accessed a URL that did not return a redirect.

Our crawler obtained two types of information for every successfully accessed URL (including those that returned a redirect): the HTML content of the page identified by the URL and the HTTP session information associated with the page request transaction. As a result, we created a file for every successfully accessed URL that contains all of this information. After our crawling process was complete, we had 348,878 Web spam pages and 223,414 redirect files (i.e., files that correspond to redirect responses). These files are collectively referred to as the Webb Spam Corpus, and they provide the basis for our analysis in this chapter.

We acknowledge that our collection of Web spam examples is not representative of all Web spam; however, it is two orders of magnitude larger than any other available source of Web spam to date, and as such, it currently provides the most realistic snapshot of Web spammer behavior. Thus, although the characteristics of our corpus might not be indicative of all Web spam, our observations still provide extremely useful insights about the techniques being employed by Web spammers.

9.2 Content Analysis

In this section, we provide the results of our large-scale analysis of Web spam content. This analysis consists of two parts. The first part, discussed in Section 9.2.1, quantifies the amount of duplication present among Web spam pages. The second part, discussed in Section 9.2.2, presents a categorization of Web spam pages.

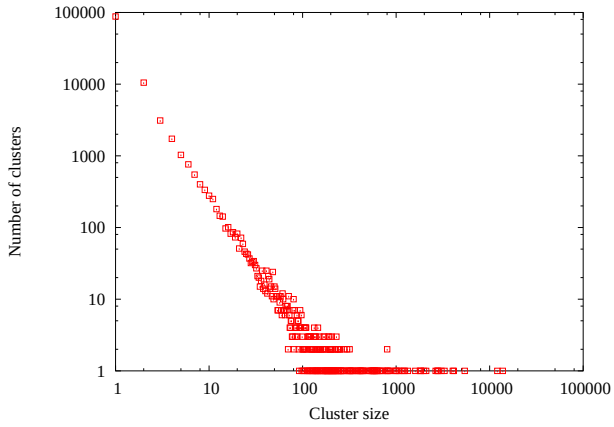


Figure 43: Number and size of the shingling clusters.

9.2.1 Web Spam Duplication

Previous research has shown that approximately one third of all Web pages are duplicates or near-duplicates of a Web page in the remaining two thirds [24, 53, 55]. To evaluate the amount of duplication among Web spam pages, we analyzed three forms of duplication in our corpus: URL duplication, content duplication, and content near-duplication.

In the previous chapter, we identified the existence of duplicate URLs in the Webb Spam Corpus (i.e., multiple Web spam pages with the same URL), and we explained that these duplicate URLs are the result of multiple unique HTTP redirect chains that lead to the same destination. Specifically, we found that the corpus contains 263,446 unique URLs, which means about one fourth of the Web spam pages have a URL that is the same as one of the Web spam pages in the remaining three fourths.

To identify content duplication, we computed MD5 hashes for the HTML content of all of the Web spam pages in our corpus. After evaluating these results, we found 202,208 unique MD5 values. Thus, 146,670 of the Web spam pages (42%) have the exact same HTML content as one of the pages in a collection of 202,208 unique Web spam pages. Many of these duplicates are explained by the URL duplication that exists in the corpus (described above), but since each of the duplicate URLs represents a distinct entry point (i.e., a unique HTTP redirect chain) to a given page, we consider them to be functionally equivalent to content duplicates.

Table 20: Most of the 50 largest equivalence classes.

Rank	Size	Categories	Most Common Domain (Count)
1	13,806	Ad Farms, Redirection	techbuyer.com (6,877)
2	12,090	Redirection	www.bizrate.com (578)
3	5,420	Pornography, Redirection	www.ezinetracking.com (832)
4	4,138	Parked Domains, Redirection	migada.com (1,553)
5, 8, 41, 46	4,034, 2,837, 594, 567	Ad Farms, Redirection	mx07.com (4,034)
6	3,294	Parked Domains, Redirection	pntaa.com (452)
7	3,053	Advertisements	www.macmall.com (3,053)
9, 10, 11, 21	2,791, 2,749, 2,646, 1,142	Ad Farms	ew01.com (6,579)
12	2,096	Advertisements	yoursmartrewards.com (2,096)
13	1,983	Parked Domains, Redirection	www.optinspecialists.info (426)
14, 32	1,837, 784	Ad farms, Redirection	click.recessionspecials.com (1,836)
15	1,828	Redirection	mailer.ebates.com (1,828)
17	1,606	Parked Domains, Redirection	www.flgstff.com (777)
18	1,336	Parked Domains	www.gibox.com (133)
19, 38	1,239, 622	Ad Farms, Redirection	lb3.netster.com (1,821)
22	1,069	Advertisements	morozware.com (62)
23, 28, 30, 34, 43, 47	1,014, 822, 802, 712, 583, 562	Ad Farms, Redirection	www.thehdhd.com (1,014)
24	995	Advertisements	www.personaloem.info (995)
25	977	Advertisements	ratedoem.info (112)
26	857	Parked Domains	pn01.com (402)
27	831	Advertisements	www.netidentity.com (831)
33	724	Parked Domains, Redirection	apps5.oingo.com (724)
35	674	Parked Domains, Redirection	www.demote.com (2)
37	630	Advertisements	www.pimsleurapproach.com (630)
42	588	Parked Domains, Redirection	new.hostcn2.com (84)
44	580	Parked Domains, Redirection	www.zudak.com (282)
45	570	Pornography	www.centerfolds4free.com (29)
48	554	Parked Domains, Redirection	landing.domainsponsor.com (542)
49, 50	536, 532	Ad Farms	dbm.consumer-marketplace.com (451)

To evaluate the amount of near-duplication in our corpus, we used the shingling algorithm that was developed by Fetterly et al. [52, 53, 55] to construct equivalence classes of duplicate and near-duplicate Web spam pages. First, we preprocessed every Web spam page in the corpus. Specifically, the HTML tags in each page were replaced by white space, and every page was tokenized into a collection of words, where a word is defined as an uninterrupted series of alphanumeric characters.

Then, for every page, we created a fingerprint for each of its n words using a Rabin fingerprinting function [124] (with a degree 64 primitive polynomial p_A). Once we had the n word fingerprints, we combined them into 5-word phrases. The collection of word fingerprints was treated like a circle (i.e., the first fingerprint follows the last fingerprint) so that every fingerprint started a phrase, and as a result, we obtained n 5-word phrases. Next, we generated n phrase fingerprints for the n 5-word phrases using a Rabin fingerprinting function (with a degree 64 primitive polynomial p_B). After we obtained the n phrase fingerprints, we applied 84 unique Rabin fingerprinting functions (with degree 64 primitive polynomials p_1, ..., p_{84}) to each of the n phrase fingerprints. For every one of the 84 functions, the smallest of the n fingerprints was stored. Once this process was complete,

each Web spam page was reduced to 84 fingerprints, which are referred to as that page's *shingles*.

Once all of the pages were converted to a collection of 84 shingles, we clustered the pages into equivalence classes (i.e., clusters of duplicate or near-duplicate pages). Two pages were considered duplicates if all of their shingles matched, and they were near-duplicates if their shingles agreed in two out of the six possible non-overlapping collections of 14 shingles. For a more detailed description of this shingling algorithm, please consult [52, 53, 55].

After running the shingling algorithm on our Web spam pages, we were left with 109,157 unique clusters of duplicate and near-duplicate pages. Figure 43 shows the distribution of the number and size of the shingling clusters. From the figure, we see that 87,819 clusters contain a single Web spam page (the point at the top-left of the figure). These pages are truly unique because none of the other pages in the corpus duplicate their content. On the opposite end of the spectrum, one cluster contains 13,806 Web spam pages (the point at the bottom-right of the figure). All of these pages are either duplicates or near-duplicates of each other. The main observation from these results is that two thirds of Web spam pages are duplicates or near-duplicates of a Web spam page in the remaining one third. Thus, duplication is twice as prevalent among Web spam pages as it is among Web pages in general.

9.2.2 Web Spam Categorization

To categorize the content of Web spam pages, we manually investigated the 50 largest equivalence classes (as defined by the clustering algorithm described in Section 9.2.1). These 50 clusters contain 93,595 Web spam pages, accounting for 26.8% of the Web spam pages in our corpus. Based on our investigation, we identified five categories that describe the pages we reviewed:

- **Ad Farms**
- **Parked Domains**
- **Advertisements**

- **Pornography**
- **Redirection**

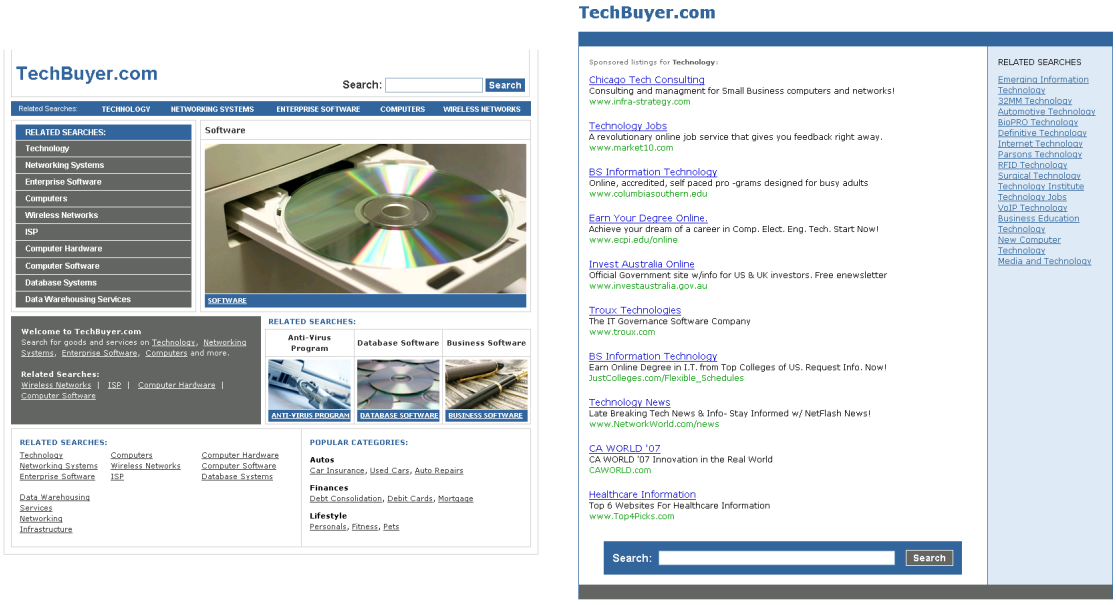
These categories help describe the purpose of the pages in each of the shingling clusters as well as the goals of the spammers who created them. Table 20 lists most of the 50 largest equivalence classes. For each cluster, we provide its rank (in terms of size), size, and categorization. We also provide the most common domain name found in each of the listed clusters. In the remainder of this section, we will describe our five Web spam categories and detail the important characteristics of their representative pages.

9.2.3 Ad Farms

Ad farms are pages that only contain advertising links (usually in the form of ad listings). These pages are of little value to visitors because they do not contain any original content. Additionally, many (if not all) of the links that appear in their ad listings are low quality because they are not ordered by traditional ranking algorithms (e.g., Google's PageRank). In fact, a large fraction of the links are controlled by the Web spammers themselves.

To deceive visitors into believing ad farms are valuable and legitimate, most Web spammers create elaborate entry pages that appear to be legitimate directories. Figure 44(a) shows an example of an ad farm's entry page. Once visitors click on one of the categories in these fake directories, they are typically redirected to an ad listing. Figure 44(b) shows an example of an ad listing that is returned when a user clicks on one of the links depicted in Figure 44(a). The links that are displayed in these ad listings are typically obtained from an ad syndicator, but the HTML structures used by ad farms are created by Web spammers.

Ad farms are extremely common in our corpus; 21 of the 50 largest equivalence classes are composed exclusively of ad farms. The 1st cluster contains pages that use JavaScript location objects and meta refresh tags to redirect users to an ad farm. Out of the cluster's 13,806 pages, only 1,821 unique hostnames are represented, and those hostnames consist of various subdomains off of 58 unique domain names (e.g., `valuevalet.seeq.com`, `happy-thoughts.seeq.com`, `inraw.seeq.com`, etc.). Spammers create numerous subdomains for three reasons. First, every subdomain represents a new address on the Web. As a result, spammers



(a) (b)

Figure 44: Ad Farm examples.

can use each of these addresses as another unique entry point into an ad farm. Second, creating multiple subdomains is far less expensive than creating an equivalent number of unique domain names. Third, the name of a given subdomain can help influence the actual advertising links that are displayed in the ad farm. Thus, spammers can maximize the coverage of their ad farms by using numerous, non-overlapping subdomain names.

The pages in the 5th, 8th, 41st, and 46th clusters use a frameset (consisting of two frames) to redirect users to an ad farm. The first frame loads fake directory content (i.e., various categories and subcategories, which lead to corresponding ad listings), and the second frame loads a search field that allows users to search for specific ad farm content. Each of the four clusters contains numerous subdomains off of a specific domain name. The 5th cluster uses `mx07.com`; the 8th cluster uses `emailcourrier.com`; the 41st cluster uses `yearendsaver.com`, and the 46th cluster uses `brightermail.com`. We grouped these clusters together in Table 20 because all of their pages have similar HTML structures and were hosted at the same IP address (64.69.68.141). The only difference between the clusters is the actual text that is used in the ad links on their pages. For example, the ad farms in the 8th cluster are primarily

concerned with email-related ads, whereas the ad farms in the 41st cluster are focused on accounting-related ads.

The 9th, 10th, 11th, and 21st clusters also contain pages that display ad farms. However, unlike the previous examples, these pages do not use redirection techniques. They display the ad farms directly. Similar to the last group of clusters, we grouped these clusters together in Table 20 because their pages use the same HTML structure, and all of their pages were hosted at one of two IP addresses (204.251.15.193 and 204.251.15.194). The clusters also use a number of subdomains off of a specific domain name. The 9th, 11th, and 21st clusters use `ew01.com`, and the 10th cluster uses `www.msstd.com`.

The 23rd, 28th, 30th, 34th, 43rd, and 47th clusters are also particularly interesting because they contain an additional level of redirection that is missing from the files in the previous clusters. First, the pages in these clusters redirect users to `lb1.youbettersearch.com`. To accomplish this initial redirection, some of the pages use the replace method for JavaScript location objects, and others use meta refresh tags. Then, that hostname uses another level of redirection to obtain the content for its ad farms.

9.2.4 Parked Domains

Domain parking services allow individuals to display a Web page that acts as a place holder for newly registered domains. A popular choice for this place holder is an ad listing because it allows an individual to monetize a domain with minimal effort. Unfortunately, Web spammers quickly exploited this opportunity and began parking hundreds of thousands of domains with ad listings.

Parked domains are functionally equivalent to ad farms. They both use ad syndicators as their primary sources of content, and they both provide little to no value to their visitors. However, parked domains possess two unique characteristics that distinguish them from ad farms. First, parked domains rely on domain parking services (e.g., `apps5.oingo.com`, `searchportal.information.com`, `landing.domainsponsor.com`, etc.) to provide their entire advertising infrastructure (the HTML structure of the entry pages as well as the content for the ad listings). Second, domain parkers are typically much more motivated than ad farmers

to sell the domains they are using to display ad links. In many cases, parked domains even include links with phrases such as "Offer To Buy This Domain" or "Purchase This Domain" to persuade visitors to buy the domain.

Eight of the clusters (#4, #6, #13, #17, #35, #42, #44, and #48) contain pages that use various techniques to redirect users to ad listings, which are provided by various domain parking services. The pages in the 4th cluster use a frameset (consisting of one frame) to redirect users to `apps5.oingo.com`, while the pages in the 6th cluster use the replace method for JavaScript location objects to accomplish the redirection. The 13th and 17th clusters consist of pages that use a frameset to redirect users to a handful of different domain parking services. For both clusters, `searchportal.information.com` is the most commonly used service. The 35th and 42nd clusters both contain pages that redirect users to `apps5.oingo.com`. The pages in the 35th cluster use a frame to accomplish the redirection, and the pages in the 42nd cluster use an iframe. The 44th cluster contains pages that use a frameset (consisting of two frames) to redirect users to `landing.domainsponsor.com` or `apps5.oingo.com`. The 48th cluster also contains pages that rely on domain parking services (`landing.domainsponsor.com` and `searchportal.information.com`). However, unlike the other clusters, which contain pages that redirect users with content-based redirection, these pages are obtained by following HTTP redirects.

Three of the clusters (#18, #26, and #33) contain pages that were generated by DNS registrars. The pages in the 18th and 26th clusters were generated by registrars that provide their own domain parking services (GoDaddy and DomainDiscover, respectively). These pages contain a combination of syndicated ad listings and registrar-specific advertisements. The pages in the 33rd cluster were generated by a DNS registrar (Network Solutions); however, the pages rely on a domain parking service (`apps5.oingo.com`) for their content.

9.2.5 Advertisements

In addition to ad farms and parked domains, which display ad listings for various Web pages, the corpus also contains numerous pages that advertise specific products and services. Ad farms and parked domains are essentially directories for advertisements, and these

(a) (b)

Figure 45: Advertisement examples.

advertisement pages are examples of the types of pages being advertised in those directories. Two examples of these advertisement pages are shown in Figures 45(a) and 45(b).

The 7th cluster contains pages that display advertisements for software and hardware products that are being sold at `macmall.com`. The pages in the 12th cluster offer free gift cards in exchange for a user's personal information (e.g., email address), and they all use the same domain name (`yoursmartrewards.com`). The 15th cluster consists of pages that use meta refresh tags to redirect users to `http://www.ebates.com`. This site is a well-known advertiser and adware/spyware distributor. The pages in the 22nd and 25th clusters all display advertisements for various software packages, and the 24th cluster contains pages that display advertisements for "Pink Floyd Products." The pages in the 27th cluster display advertisements for various domain names being sold by `www.netidentity.com`. These pages are different from parked domains because the pages are not concerned with generating ad revenue – their sole purpose is selling domains. The 37th cluster contains pages that display advertisements for various foreign language instructional materials (e.g., books, tapes, videos, etc.) being sold at `www.pimsleurapproach.com`.

9.2.6 Pornography

Although only 2 of the 50 largest equivalence classes consist of pornography-related pages, those 2 clusters account for almost 2% of the entire corpus. The 3rd cluster contains 5,420 pages that execute "drive-by advertising." Specifically, these pages generate pop-up advertisements for `www.freeezinebucks.com`, and then, they use meta refresh tags to redirect users to various pornographic Web sites (e.g., `www2.grandegirls.com`, `www.wannawatch.com`, etc.). The 45th cluster contains 570 pages that prompt the user to log in to a pornographic site (e.g., `www.brunettes4free.com`, `www.girlgirl4free.com`, etc.).

9.2.7 Redirection

Many Web spammers use redirection to hide their spam content [75]. Thus, one of the most ubiquitous characteristics of Web spam pages is their use of redirection. Table 20 shows that 27 of the 50 largest equivalence classes contain pages that utilize redirection techniques. In this section, we investigate the most popular techniques, and we present the most popular redirection destinations.

The easiest way to accomplish redirection is at the HTTP-level (i.e., returning a 3xx status code). As explained in the previous chapter, the Webb Spam Corpus contains 223,414 redirect files that represent examples of this type of HTTP redirection. All of these HTTP redirect files contain one of two 3xx status codes: 301 ("Moved Permanently") and 302 ("Found"). Aside from HTTP redirection, a number of content-based redirection techniques also exist. Based on our manual examination of the largest equivalence classes in the corpus, we identified six content-based redirection techniques that are repeatedly employed by Web spammers. Three of these techniques are accomplished using HTML, and the other three are accomplished using JavaScript. The HTML techniques make use of meta refresh tags, frame tags, and iframe tags. The JavaScript techniques include assigning a URL to a location object, assigning a URL to a location object's href attribute, and passing a URL to the replace method of a location object.

Identifying examples of the HTML redirection techniques was fairly straightforward due to the syntactic properties of the HTML tags that are used (meta, frame, and iframe).

Specifically, we wrote a custom HTML parser (based on Perl's HTML::Parser module) to identify these tags and extract the URLs being used for redirection. For the remainder of this chapter, we will refer to these extracted URLs (and their corresponding hostnames) as *targets* of redirection.

Identifying examples of the JavaScript redirection techniques was significantly more challenging for a number of reasons. First, many of the pages in the corpus contain external JavaScript references that use relative addresses. These relative addresses rely on the existence of locally stored JavaScript scripts. However, the files in the corpus only contain HTML content and embedded JavaScript scripts (i.e., none of the external JavaScript scripts are stored locally). To solve this problem, we dynamically rewrote the HTML files, replacing the relative script addresses with absolute addresses. As a result, we were able to download the necessary external script files when they were needed.

Another challenge posed by the JavaScript techniques is the nondeterministic behavior of JavaScript script execution. Unlike the HTML techniques, which we easily identified with an HTML parser, the JavaScript techniques were often hidden by conditional statements or accomplished with additional levels of indirection (e.g., method calls). Additionally, many of the techniques used variables to assign the targets of redirection (as opposed to using direct assignments).

To overcome these obstacles, we dynamically rewrote the HTML files to trap JavaScript method calls (e.g., replace()) and assignments to important JavaScript objects and attributes (e.g., location and location.href) that dealt with redirection. Specifically, we replaced each redirection technique with an alert method. Then, to capture the targets of redirection, we passed the original redirection parameters as arguments to the alert method. As a result, the JavaScript redirection techniques were replaced as follows:

- *location = URL;* became *alert("location: URL");*
- *location.href = URL;* became *alert("location.href: URL");*
- *location.replace(URL);* became *alert("location.replace: URL");*

In each of the above replacements, *URL* could be a static string or a variable construction. Our HTML rewriting techniques were able to handle both of these cases.

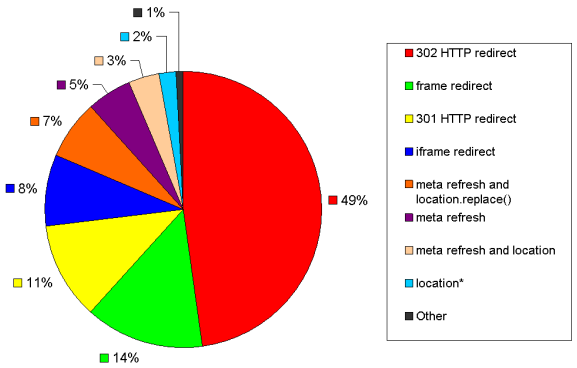


Figure 46: Relative frequency of redirection techniques.

After rewriting our corpus files, we used HtmlUnit 1.10[1] to create a custom WebClient that trapped alert method calls and parsed their arguments (i.e., the targets of redirection). Then, we used our WebClient to access the rewritten HTML files, execute their JavaScript scripts (using the Rhino JavaScript engine[2]), and capture the alert method calls that were generated by JavaScript redirection techniques. Finally, we extracted the redirection targets and converted any relative target addresses to absolute target addresses.

Based on our analysis, we discovered that the corpus contains 144,801 unique redirect chains, each containing an average of 1.54 HTTP redirects. Thus, 41.5% of the Web spam pages were obtained by following a redirect chain. Additionally, of the 348,878 Web spam pages, 153,265 (43.9%) use some form of HTML or JavaScript redirection. Only 1,304 of the 223,414 redirect files (0.6%) use HTML or JavaScript redirection techniques, but that is not surprising since most of those files only contain session information for HTTP redirects.

Figure 46 shows a pie chart that breaks down the relative frequency of the redirection techniques across all of the corpus files (i.e., Web spam pages and redirect files). HTTP redirection (without the use of any content-based techniques) is clearly the most popular form of redirection in the corpus, accounting for 60% of the redirections (49% for "Found" redirects and 11% for "Moved Permanently" redirects). The next most popular techniques involve only using HTML frame tags or HTML iframe tags. These techniques account for

[1] `http://htmlunit.sourceforge.net/`
[2] `http://www.mozilla.org/rhino/`

Table 21: Most common targets of redirection.

Top 5 targets of redirection	
Hostname	Count
lb1.youbettersearch.com	44,334
ads2.drivelinemedia.com	15,798
bluerocketonline.TechBuyer.com	12,204
login.tracking101.com	11,639
apps5.oingo.com	10,153
Top 5 targets of HTTP redirection	
Hostname	Count
login.tracking101.com	11,639
www.macmall.com	8,895
cpaempire.com	5,350
mailer.ebates.com	3,656
click.be3a.com	2,488
Top 5 targets of frame redirection	
Hostname	Count
apps5.oingo.com	6,952
searchportal.information.com	5,796
landing.domainsponsor.com	5,256
click.recessionspecials.com	1,836
migada.com	1,553
Top 5 targets of iframe redirection	
Hostname	Count
ads2.drivelinemedia.com	15,798
simg.zedo.com	6,002
lb3.netster.com	4,961
apps5.oingo.com	3,201
www.creativecow.net	2,098
Top 5 targets of meta refresh redirection	
Hostname	Count
lb1.youbettersearch.com	22,278
bluerocketonline.TechBuyer.com	6,108
www.ebates.com	1,828
biz.tigerdirect.com	803
programs.weginc.com	722
Top 5 targets of location* redirection	
Hostname	Count
lb1.youbettersearch.com	22,056
bluerocketonline.TechBuyer.com	6,096
www.classmates.com	659
yoursmartrewards.com	427
c.azjmp.com	417

14% and 8% of the redirections, respectively. Redirection using meta refresh tags appears in a variety of flavors. The most popular form of meta refresh redirection is accomplished in conjunction with the replace method of a JavaScript location object. This technique accounts for 7% of the redirections in the corpus. Files that exclusively use one of the three JavaScript techniques, which we grouped together as "location*" in the figure, account for 2% of the redirections. All of the other combinations of redirection techniques collectively account for 1% of the redirections.

Table 21 shows the hostnames that are most frequently the targets of redirection in our corpus. The first set of counts represent the combined view of all of the HTTP, HTML,

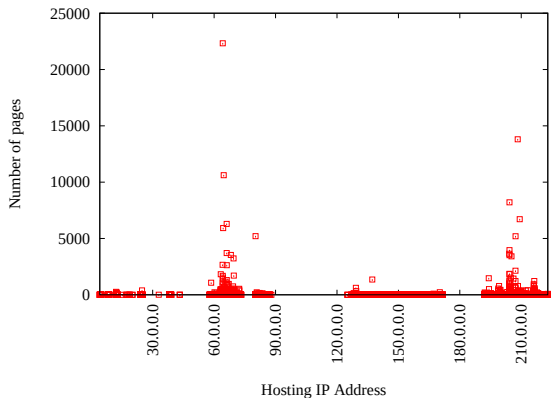


Figure 47: Number of pages being hosted by a single IP address.

and JavaScript redirection techniques we identified. This list consists of 2 ad farms (`lb1.youbettersearch.com` and `bluerocketonline.TechBuyer.com`), 2 advertisers (`ads2.drivelinemedia.com` and `login.tracking101.com`), and 1 domain parking service. The top 5 HTTP redirect targets are all advertisers. The top 5 frame redirect targets consist of 3 domain parking services (`apps5.oingo.com`, `searchportal.information.com`, and `landing.domainsponsor.com`), 1 ad farm (`click.recessionspecials.com`), and 1 parked domain. The top 5 iframe redirect targets consist of 1 ad farm (`lb3.netster.com`), 3 advertisers (`ads2.drivelinemedia.com`, `simg.zedo.com`, and `www.creativecow.net`), and 1 domain parking service (`apps5.oingo.com`). The top 5 meta refresh redirect targets consist of 2 ad farms (`lb1.youbettersearch.com` and `bluerocketonline.TechBuyer.com`), 2 advertisers (`www.ebates.com` and `biz.tigerdirect.com`), and 1 pornographer. The top 5 location* redirection targets consist of 2 ad farms (`lb1.youbettersearch.com` and `bluerocketonline.TechBuyer.com`) and 3 advertisers (`www.classmates.com`, `c.azjmp.com`, and `yoursmartrewards.com`).

9.3 HTTP Session Analysis

In addition to the HTML content of Web spam pages, our corpus also contains the HTTP session information that was obtained from the servers that were hosting those pages. In this section, we characterize this session information, focusing on the most common server IP addresses and session header values.

Table 22: Top 10 hosting IP addresses.

Hosting IP Address	Count
64.225.154.135	22,332
208.254.3.166	13,806
64.69.68.141	10,615
204.251.15.194	8,211
209.233.130.40	6,713
66.116.109.62	6,294
64.40.102.44	5,923
80.245.197.244	5,206
207.219.111.23	5,201
204.251.15.193	3,963

9.3.1 Hosting IP Addresses

One of the most important pieces of HTTP session information is the IP address that hosted a given Web spam page – the *hosting IP address*. Figure 47 shows the distribution of all of the hosting IP addresses in our corpus. This figure clearly shows that most of the hosting IP addresses were concentrated around a few IP address ranges. Specifically, the 63.* – 69.* and 204.* – 216.* IP address ranges account for 45.4% and 38.6% of the hosting IP addresses in the corpus, respectively (84%, collectively). Table 22 shows the 10 most popular hosting IP addresses, and all but one of them are in these IP address ranges.

Interestingly, these IP address ranges also include the two most popular Web spam IP address ranges discussed in previous work by Wang et al. [150]. Specifically, they found that most of their spam examples were being hosted on two IP address ranges (64.111.* and 66.230.*). Our corpus includes 120 and 916 examples from those address ranges, respectively. The presence of these hosting IP addresses in our corpus reaffirms their results, and it also emphasizes the value of our corpus and the method used to obtain it.

9.3.2 HTTP Session Headers

In addition to the hosting IP addresses of Web spam pages, our corpus also contains all of the HTTP session headers that were associated with the page request transaction. In this section, we identify the most commonly used headers as well as the most popular header

Table 23: Top 10 HTTP session headers.

Header	Total Count	Unique Count	Most Popular Value (Count)
Content-Type	348,878	688	text/html (155,401)
Server	343,168	6,513	microsoft-iis/6.0 (64,787)
Connection	327,478	6	close (304,557)
X-Powered-By	209,215	261	asp.net (80,294)
Content-Length	162,532	31,232	1470 (6,115)
Cache-Control	148,715	548	private (69,571)
Set-Cookie	145,315	140,431	gx_jst=9fa7274e662d6164; path=/apps/system, gx_jst=9fa7274e662d6164 (626)
Link	142,785	15,573	</style/kentech.css>; rel="stylesheet"; type="text/css" (25,620)
Expires	93,477	25,056	mon, 26 jul 1997 05:00:00 gmt (18,933)
Pragma	75,435	32	no-cache (64,344)

values.

Many of the corpus files contained more than one value for a given header. In each of those cases, we concatenated all of the separate values into a comma delimited list. We did this because we wanted a one-to-one mapping for a file and each of its headers to simplify header comparisons from one file to another.

Table 23 shows the 10 most popular HTTP session headers in terms of the number of Web spam pages that contain them. The table shows the number of pages each header appears in, the number of unique values each header has, and the most popular value for each of the headers. The "Content-Type" header is the most popular header, appearing in all 348,878 of the Web spam pages in our corpus. As we explained in the previous chapter, the corpus only contains files with textual "Content-Type" values. Thus, the values for the "Content-Type" header are primarily "text/html" combined with permutations of various charset encodings (e.g., "iso-8859-1," "utf-8," etc.).

The "Server" header is the second most popular header, appearing in 343,168 of the Web spam pages. This header is extremely important because it describes the Web server that actually served a given Web spam page. Microsoft IIS 6.0 is the most commonly used Web server in our corpus (18.9% of the pages were hosted by it), but generally, Apache (63.9%) was used more frequently than Microsoft IIS (30.3%). Overall, these two Web servers were clearly the most popular option among Web spammers, accounting for 94.2% of the pages that contain a "Server" header.

9.4 Related Work

Fetterly et al. [54] statistically analyzed two data sets of Web pages (DS1 and DS2) using properties such as linkage structure, page content, and page evolution. They found that many of the outliers in the statistical distributions of these properties were Web spam, and they manually identified 98 out of 1,286 Web pages as spam. Ntoulas et al. [115] extended the work by investigating additional content-based features of a collection of 2,364 Web spam pages (e.g., fraction of visible content, compressibility, independent n-gram likelihoods, etc.). Castillo et al. [27] also identified a few spam features (e.g., synthetic text, parked domains, etc.) using a collection of 1,447 manually labeled Web spam pages. Our work differs from these previous evaluations in two very important ways. First, our characterization was performed on a collection of Web spam pages that is two orders of magnitude larger than the collections used in previous studies. Second, we are the first to analyze the HTTP session information associated with Web spam pages.

Wu and Davison [153] performed a preliminary evaluation of the redirection techniques used on the Web. Specifically, they looked at HTTP redirection, meta refresh redirection, and two types of JavaScript redirection in a collection of unlabelled Web pages. Wang et al. [150] took this work a step further by analyzing network redirection traffic from known spam domains to identify redirection URLs. In this chapter, we provide an evaluation of the redirection techniques used by Web spammers that enhances these previous studies in three ways. First, our analysis encompasses hundreds of thousands of Web spam pages, whereas previous studies only used a few thousand pages. Second, we investigate a more comprehensive list of redirection techniques: HTTP redirection, 3 HTML-based techniques, and 3 JavaScript-based techniques. Third, previous research [75, 153] detailed the difficulties associated with identifying and processing JavaScript redirection techniques. We were able to overcome these difficulties by using sophisticated HTML rewriting techniques, and as a result, we are the first to present a large-scale evaluation of JavaScript-based redirection techniques.

Wang et al. [149] developed an automated approach for identifying typo-squatting domains, which are a specific type of parked domains. Using this approach, they found that

a handful of domain parking services are responsible for parking about 30% of the typo-squatting domains they identified. In our study, we also identified numerous examples of parked domains; however, our analysis was not confined to typo-squatting domains, and we investigated a significantly larger collection of Web spam pages.

9.5 Summary

We have conducted the first large-scale experimental study of Web spam through content and HTTP session analysis on the Webb Spam Corpus – a collection of almost 350,000 Web spam pages. Our results are consistent with the hypothesis that Web spam pages are fundamentally different from normal Web pages. Specifically, we found that the rate of duplication among Web spam pages is twice the duplication rate for normal Web pages. Our content analysis also found five important categories of Web spam: **Ad Farms**, **Parked Domains**, **Advertisements**, **Pornography**, and **Redirection**.

In addition to content analysis, we also performed HTTP session analysis on the session data that was collected during the construction of the Webb Spam Corpus. This session analysis showed two trends. First, the hosting IP addresses are concentrated in two narrow ranges (63.* – 69.* and 204.* – 216.*). Second, significant overlaps exist among the session header values. Both of these trends are consistent with the hypothesis that Web spam pages are detectably different from normal Web pages, in a way similar to the results of our content analysis.

Although previous Web spam research has focused primarily on link analysis, our results suggest that content and HTTP session analysis techniques can contribute greatly towards distinguishing Web spam pages from normal Web pages. This is a promising result because these techniques can become new weapons in our ongoing battle against Web spam.

CHAPTER X

PREDICTING WEB SPAM WITH HTTP SESSION INFORMATION

The use of malicious Web pages to influence human users and attack browser client machines has grown significantly despite continued research and industry efforts. Examples of malicious Web pages (commonly called Web spam) include pages that subvert search engine ranking algorithms (accounting for between 13.8% and 22.1% of all Web pages [27, 115]) as well as pages designed to propagate malware by hosting Web-borne spyware and browser exploits [111, 112, 121, 149]. Thus, for both performance and security reasons, the identification of Web spam pages has become increasingly important.

Most previous and current research efforts on Web spam defenses have focused on protecting human users by identifying Web pages that skew search engine rankings through link or content manipulations. Representative examples of this research include link-based [15, 17, 28, 30, 76, 154] and content-based [54, 115] analysis techniques. Although these analysis techniques effectively identify certain types of Web spam, the techniques need to download and inspect the content associated with suspect Web pages for the analysis algorithms to work. This dependence on Web spam content makes these previous approaches practically defenseless against Web-propagated malware, and it severely limits their impact on the resource burdens imposed by Web spam.

Notwithstanding the common assumption that Web spam content is necessary for identifying Web spam, the main hypothesis of this chapter is that HTTP session information (i.e., hosting IP addresses and HTTP session headers) provides sufficient evidence for the successful identification of many Web spam pages. This hypothesis is supported by the results in the previous chapter, which show that very distinct patterns emerge within the HTTP session information associated with Web spam pages. In this chapter, we leverage that observation by using HTTP session information to train classification algorithms to distinguish between spam and legitimate Web pages.

The first contribution of this chapter is a new approach for retrieving Web pages using HTTP. Specifically, we insert an HTTP session classifier into the HTTP retrieval process, which predicts whether a Web page is spam based on the page's HTTP session information (completely ignoring its content). This predictive approach protects Web users from Web-propagated malware, and it generates significant bandwidth and storage savings because it avoids downloading and storing useless Web spam pages. Our approach is particularly useful for search engines because it enables more efficient and effective Web crawlers, which will generate indexes with higher quality content.

The second contribution of this chapter is a large-scale experimental evaluation of Web spam classification using HTTP session information. As part of this evaluation, we investigate various classification algorithms and discover that many classifiers are capable of successfully detecting almost 90% of the Web spam in our corpora. Then, we evaluate the effects of varying class distributions and observe that our best classifiers are very robust under a number of environmental circumstances. Finally, we investigate the computational costs and resource savings associated with our approach. Based on our experiments, we observe that our most effective classifier only adds an average of 101μs to each Web page retrieval, while saving an average of 15.4 KB of bandwidth and storage resources for every successfully identified Web spam page.

The remainder of the chapter is organized as follows. Section 10.1 reviews previous research on Web spam identification and classification. Section 10.2 provides background information about HTTP operations and describes our new approach for retrieving Web pages. Section 10.3 describes the experimental setup we used to evaluate our new approach, and Sections 10.4 and 10.5 present our experimental results using various corpora of spam and legitimate Web pages. We summarize our findings in Section 10.6.

10.1 Related Work

Previous Web spam research can be categorized broadly into three groups: link-based, content-based, and hybrid analysis techniques. Link-based analysis techniques attempt to identify Web spam pages based on their hyperlink connections to other pages. Davison [43]

was the first to investigate link-based Web spam, building decision trees to successfully identify "nepotistic links." Becchetti et al. [15] revisited the use of decision trees on a newer collection of Web data and were able to successfully identify 80.4% of the 840 Web spam examples in their sample with a false positive rate of 1.1%. To help mitigate the effects of link manipulations on link-based ranking algorithms, Gyöngyi et al. [76] developed an algorithm called TrustRank that can be used to propagate trust from a seed set of Web pages to the successors of those pages. Wu and Davison [154] and Benczur et al. [17] also proposed solutions for improving the resiliency of ranking algorithms. However, instead of propagating trust to good pages, they propagated distrust to bad pages. Finally, Caverlee et al. [30] proposed a spam resilient ranking approach called Spam-Resilient SourceRank. This approach collects Web pages into logical groupings called sources, and it provides parameters that can be tuned to throttle the ranking influence of various sources.

Content-based analysis techniques focus on identifying fundamental differences between the content of spam and legitimate Web pages. Fetterly et al. [54] statistically analyzed two data sets of Web pages (DS1 and DS2) using properties such as page content, content duplication, and page evolution. They found that many of the outliers in the statistical distributions of these properties were Web spam, and they manually identified 98 out of 1,286 Web pages as spam. Ntoulas et al. [115] extended the work by using additional content-based features (e.g., fraction of visible content, compressibility, independent n-gram likelihoods, etc.) to build decision trees, which were able to successfully classify 86.2% of their collection of 2,364 spam pages with a false positive rate of 1.3%. Anderson et al. [5] developed a technique called spamscatter, which extracts URLs from spam email messages, obtains the corresponding Web content, and clusters that Web content based on an image shingling technique. Using their approach on a one-week collection of email, they were able to identify 2,334 unique "scams" being hosted on 7,029 unique IP addresses.

Hybrid analysis techniques combine content-based and link-based techniques to identify Web spam. Gyöngyi and Garcia-Molina [75] proposed a taxonomy that categorizes various techniques used by Web spammers to manipulate search engine results. This taxonomy showcases a variety of content-based and link-based spamming techniques. Drost

Request Line	`GET / HTTP/1.1`
Headers	`Host: click.recessionspecials.com` `User-Agent: Mozilla/5.0`
Empty Line	

Figure 48: Example HTTP request message.

and Scheffer [45] trained a support vector machines (SVM) classifier using content-based features (e.g., number of page tokens, number of URL characters, etc.) as well as link-based features (e.g., contextual similarities between a page, its predecessor, and its successors). Then, using a private collection of 1,285 Web pages (431 spam and 854 legitimate), their classifier consistently obtained area under the ROC curve (AUC) values greater than 90%. Castillo et al. [28] created a cost-sensitive decision tree using the content-based features described in [115] and link-based features described in [15]. Their decision tree was able to successfully identify 88.4% of their 675 Web spam examples with a false positive rate of 6.3%. Finally, Wang et al. [150] proposed a five-layer double-funnel model for analyzing Web spam. Then, using this model and the Strider Search Ranger system, which combines content-based and redirection traffic analysis techniques, they identified various URLs and IP addresses that were associated with Web spam pages.

10.2 Web Page Retrieval Using HTTP

In this section, we review the underlying protocols responsible for Web browsing and Web crawling. First, we discuss the HTTP operations available for retrieving a Web page from a Web server. Then, we propose a new approach for retrieving Web pages that leverages HTTP session classification to avoid downloading Web spam content.

10.2.1 Traditional Approach

The HyperText Transfer Protocol (HTTP) [56] is a request/response protocol that allows clients (or user agents) to exchange information with servers (or origin servers) located around the world. In the Web context, user agents are typically Web browsers and Web crawlers, and origin servers are typically Web servers that store various forms of Web data.

The information exchange between user agents and origin servers is accomplished by transmitting MIME-like messages (as specified in RFC 2616 [56]) back and forth between

Status Line	HTTP/1.1 200 OK
Headers	Connection: close Date: Fri, 23 Dec 2005 18:12:19 GMT Server: Apache/2.0 Content-Length: 732 Content-Type: text/html; charset=UTF-8 Link: <http://static.hitfarm.com/template/qing/images/qing.ico>; /="/"; rel="shortcut icon"; type="image/x-icon" P3P: CP="NOI COR NID ADMa DEVa PSAa PSDa STP NAV DEM STA PRE" Set-Cookie: source=1; expires=Fri, 23 Dec 2005 20:12:19 GMT Title: recessionspecials.com X-Powered-By: PHP/5.0.5
Empty Line	
Message Body	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <html lang="en"> <head> <title>recessionspecials.com</title> <link rel="shortcut icon" href="http://static.hitfarm.com/template/ qing/images/qing.ico" type="image/x-icon" /> </head> <frameset cols="1,*" border="0" frameborder="0"> <frame name="hftop" src="/top.php" scrolling="no" frameborder="0" marginwidth="0" marginheight="0" noresize="noresize" /> <frame name="hfasi" src="http://apps5.oingo.com/apps/domainpark/ domainpark.cgi?cid=MEDI3409&s=recessionspecials.com& ip=130.207.5.18" scrolling="auto" frameborder="0" marginwidth="0" marginheight="0" noresize="noresize" /> <noframes> <body> This page requires frames </body> </noframes> </frameset> </html>

Figure 49: Example HTTP response message.

the user agents and origin servers. First, the user agent initiates a request by sending a *request message* to the origin server. A request message contains a request line, zero or more session headers, an empty line that denotes the end of the headers, and an optional message body. The request line includes a request method, a URL on which to apply the request method, and the user agent's supported HTTP version (HTTP/1.1 is the most widely supported version). An annotated example of a simple "GET" request message is shown in Figure 48.

After the origin server receives a request message, the server performs the requested method on the provided URL and returns the result in a *response message*. Figure 49 shows the annotated response message that corresponds to the "GET" request message shown in Figure 48. A response message contains a status line, zero or more headers, an empty line that denotes the end of the headers, and an optional message body. The status line includes the origin server's supported HTTP version, a three digit status code (e.g., "200") that indicates whether or not the request was successful, and a textual description of the status code (e.g., "OK").

10.2.2 Proposed Approach

To improve the Web browsing experience for users and to enhance the quality of information obtained by Web crawlers, we propose a new approach for retrieving Web pages with HTTP. Our proposed approach does not affect the underlying HTTP operations; however, it does change the manner in which user agents respond to the results of those operations. Specifically, we insert an HTTP session classifier into the retrieval process to dramatically reduce the amount of Web spam that is retrieved by user agents. First, the user agent initiates a "GET" request using the approach described above in Section 10.2.1. Upon receiving the request message, the origin server performs the requested method and returns the result in a response message.

In the traditional approach, the user agent blindly accepts the response (and its corresponding Web page), continuing to its next task (i.e., crawling the contents of the next URL, requesting images or other embedded objects that are found in the received Web page, etc.). However, in our proposed approach, the user agent only reads the response line and HTTP session headers from the response message (i.e., it reads up to the empty line). Then, the user agent employs a classifier to evaluate the headers and classify them as spam or legitimate. If the headers are classified as spam, the user agent closes its connection with the origin server, ignoring the remainder of the response message and saving valuable bandwidth and storage resources. Alternatively, if the headers are classified as legitimate, the user agent finishes reading the response message and continues its normal operation.

Our new approach offers at least three benefits. First, the approach is broadly applicable to any URL on the Web, regardless of where that URL is obtained (e.g., in an email, in a search engine result, in a social networking community, etc.). Since we integrate our predictive technique into the HTTP retrieval process, we are able to protect all of the page requests made by a user agent. Second, by making the classification decision at the HTTP level, we avoid downloading the content of a page unless we predict the page is legitimate. Thus, our approach enables more efficient and effective Web crawlers by providing significant bandwidth and storage savings. Additionally, by restricting content downloads, we can protect Web users against exposure to Web-borne spyware and browser exploits, which are

becoming increasingly prevalent on Web spam pages [111, 121, 149]. Third, our approach is complementary to existing link-based and content-based analysis techniques. Hence, the approach can be combined with existing techniques to create a multi-layered defense against Web spam.

10.3 Experimental Setup

The success of our predictive approach is contingent upon the performance of HTTP session classification. Therefore, we performed extensive evaluations to determine the effectiveness of classifying Web spam using HTTP session information. In this section, we discuss the fundamental principles used to perform our experimental evaluations. Then, in Sections 10.4 and 10.5, we provide experimental evidence that HTTP session classification is an effective and efficient solution for automatically predicting Web spam pages.

10.3.1 Classification Framework

Previous research [15, 28, 43, 45, 115] has shown that Web spam detection can be modeled as a binary classification problem, where the two classes are spam and legitimate (or non-spam). In binary classification problems, a model (or classifier) is built and evaluated in two separate phases: the training phase and the classification phase (or testing phase). During the training phase, a set of labeled instances (i.e., the training data) is used to train a classifier, establishing a mapping between instance features and the classes. During the classification phase, the trained classifier is used to classify a separate set of labeled instances (i.e., the testing data), and the resulting classifications are presented in the form of a confusion matrix (or contingency table):

		Predicted Class	
		Legitimate	Spam
Correct Class	Legitimate	a	b
	Spam	c	d

In the above confusion matrix, a represents the number of correctly classified legitimate Web pages, b represents the number of legitimate Web pages that were misclassified as spam, c represents the number of Web spam pages that were misclassified as legitimate,

and d represents the number of correctly classified Web spam pages. Once a confusion matrix is generated, we compute various performance metrics to evaluate the effectiveness of a classifier: true positive (TP) rate, false positive (FP) rate, F-measure, and Accuracy. The TP rate is the same as recall R, which is $\frac{d}{c+d}$, and the FP rate is $\frac{b}{a+b}$. F-measure is defined as $\frac{2PR}{P+R}$, where P is $\frac{d}{b+d}$, and Accuracy is defined as $\frac{a+d}{a+b+c+d}$. All of these metrics are well known and widely cited throughout previous machine learning and information retrieval research.

To improve the reliability of our classifier evaluations, each evaluation was performed using stratified tenfold cross-validation [96]. In stratified tenfold cross-validation, a fixed sample of data is randomly divided into ten equally-sized folds (or partitions), and each fold is comprised of approximately the same class distribution as the original sample. After the data is split, each fold is evaluated with a classifier that was trained with the other nine folds, and a confusion matrix is generated by averaging the results of these ten evaluations. Finally, the performance metrics are calculated using this confusion matrix.

10.3.2 Corpora

In Chapter 8, we developed an automatic technique for obtaining Web spam examples that leverages the presence of URLs in email spam messages. Specifically, we extracted almost 1.2 million unique URLs from more than 1.4 million email spam messages. Then, we built a crawler to obtain the Web pages that corresponded to those URLs. Our crawler obtained two types of information for every successfully accessed URL: the HTML content of the page identified by the URL and the HTTP session information associated with the page request transaction. After our crawling process was complete, we had 348,878 Web spam pages, which are referred to as the Webb Spam Corpus. This corpus is currently the largest publicly available collection of Web spam, and it served as our primary source of spam instances.

In addition to the Webb Spam Corpus, we also used the labeled spam instances in the WEBSPAM-UK2006 corpus[1] as a source of Web spam pages. The WEBSPAM-UK2006

[1]The WEBSPAM-UK2006 corpus can be found at `http://www.yr-bcn.es/webspam/datasets/`.

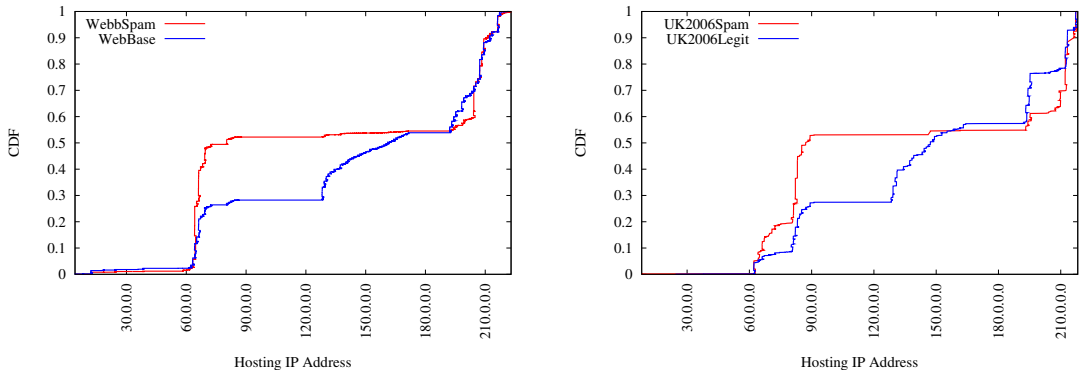
Table 24: Corpora summaries.

Corpus Abbreviation	Number of Instances	Crawl Date
WebbSpam	348,878	December 2005
WebBase	392,664	December 2005
UK2006Spam	1,338	July 2007
UK2006Legit	3,542	July 2007

corpus [27] is a publicly available Web spam collection that is based on a May 2006 crawl of the .uk domain. This corpus is much smaller and far less diverse than the Webb Spam Corpus, but since the WEBSPAM-UK2006 corpus has appeared in recent Web spam classification research [15, 28], we used it for comparative purposes. Unfortunately, this corpus does not contain HTTP session information, and as a result, we were forced to recrawl its spam URLs and obtain their corresponding HTTP session information. We performed this crawl in July 2007 (i.e., a little over a year after the original corpus was created), and we were only able to obtain 1,338 out of the 1,924 spam pages (70%) in the corpus because the missing pages were no longer available.

Since the Webb Spam Corpus was collected at the end of December 2005, we needed legitimate Web data from a similar time period to establish a fair comparison between spam and legitimate HTTP session information. To obtain this temporally similar data, we used the December 2005 crawl from Stanford's publicly available WebBase Web Page Repository [82]. This crawl is comprised of 20.7 million pages, which are stored along with the HTTP session information that was returned when the pages were crawled. We extracted a stratified random sample of 392,664 legitimate pages from this crawl, maintaining the same relative distribution of hosts in our sample.

In addition to our WebBase data, we also used the labeled legitimate instances in the WEBSPAM-UK2006 corpus as a source of legitimate Web data. As mentioned above, this corpus does not contain HTTP session information; thus, we recrawled the legitimate URLs from the corpus to obtain that data. We performed this crawl in July 2007, and we were only able to obtain 3,542 out of the 5,549 legitimate pages (64%) in the corpus because the



(a) WebbSpam and WebBase

(b) UK2006Spam and UK2006Legit

Figure 50: Hosting IP addresses. In (a), we compare the hosting IP addresses from the Webb Spam Corpus and WebBase data. In (b), we compare the hosting IP addresses for the spam and legitimate pages in the WEBSPAM-UK2006 corpus.

missing pages were no longer available.

Table 24 presents a summary of our corpora. For each corpus, the table lists its abbreviated name, its number of instances, and the date it was created.

10.3.3 HTTP Session Information Features

Classification algorithms require their inputs (or data instances) to be represented in a consistent format. In our experiments, we adopted the traditional vector space model [129] (or "bag of words" model), which has been quite effective in previous information retrieval and machine learning research. In the vector space model, each data instance is represented as a feature vector $\mathbf{f}$ of n features: $< f_1, f_2, \ldots, f_n >$. All of our features are Boolean; hence, if $f_i = 1$, the feature is present in a given instance; otherwise, the feature is absent.

Unlike previous Web spam classification research, which has relied on link-based and content-based analysis techniques, our work focuses exclusively on HTTP session information. One of the most important pieces of HTTP session information is the IP address of the Web server that hosts a given Web page – the *hosting IP address*. Figure 50(a) compares the distributions of the hosting IP addresses for the Webb Spam Corpus and WebBase data. The figure clearly shows that each distribution is quite distinct. The hosting IP addresses in the Webb Spam Corpus are concentrated around a few IP address ranges. Specifically,

Table 25: Most popular HTTP session headers.

Header	WebbSpam Total Count (Unique Count)	WebBase Total Count (Unique Count)	UK2006Spam Total Count (Unique Count)	UK2006Legit Total Count (Unique Count)
Content-Type	348,878 (688)	392,600 (122)	1,338 (41)	3,542 (123)
Server	343,168 (6,513)	387,301 (2,505)	1,336 (196)	3,490 (704)
Connection	327,478 (6)	354,837 (8)	1,322 (2)	3,230 (4)
X-Powered-By	209,215 (261)	114,181 (111)	862 (52)	1,753 (90)
Content-Length	162,532 (31,232)	214,100 (58,417)	498 (447)	2,203 (1,987)
Cache-Control	148,715 (548)	103,916 (1,461)	283 (27)	1,377 (117)
Set-Cookie	145,315 (140,431)	110,519 (105,636)	287 (287)	1,392 (1,376)
Link	142,785 (15,573)	79 (64)	797 (421)	2,690 (2,215)
Expires	93,477 (25,056)	55,991 (31,197)	183 (44)	745 (353)
Pragma	75,435 (32)	29,464 (9)	167 (5)	541 (10)
Last-Modified	73,071 (50,206)	149,191 (125,573)	418 (380)	1,348 (1,245)
P3P	59,023 (816)	21,970 (205)	27 (14)	84 (54)
Accept-Ranges	47,821 (4)	157,384 (4)	442 (1)	1,245 (2)
X-Meta-Robots	45,900 (241)	N/A	327 (25)	662 (64)
Refresh	45,075 (8,556)	136 (6)	9 (8)	150 (137)
Etag	42,546 (24,783)	118,804 (115,122)	347 (317)	1,125 (1,039)
Vary	25,721 (45)	15,939 (27)	43 (5)	120 (12)
Content-Language	20,397 (163)	9,661 (49)	138 (7)	401 (18)

the 63.* – 69.* and 204.* – 216.* IP address ranges account for 45.4% and 38.6% of the addresses associated with that corpus, respectively (84%, collectively). On the other hand, only 50.6% of the WebBase data's hosting IP addresses are in those ranges (21.8% for 63.* – 69.* and 28.8% for 204.* – 216.*). Similarly, 25.6% of the hosting IP addresses associated with the WebBase data are in the 128.* – 171.* range, whereas only 2.3% of the addresses in the Webb Spam Corpus fall within that range. Figure 50(b) compares the distributions of the hosting IP addresses for the spam and legitimate instances in the WEBSPAM-UK2006 corpus, showing distinct trends that are similar to those described for the Webb Spam Corpus and WebBase data. Since the distributions of spam and legitimate hosting IP addresses are quite distinct, we used those addresses as features in our classification experiments to help distinguish between spam and legitimate Web pages.

In addition to hosting IP addresses, we also derived a number of features from HTTP session header values. Table 25 shows a list of the most popular HTTP session headers in our corpora. For each corpus, the table lists the number of pages associated with each header and the number of unique values each header has in the corpus. Two important observations emerge from this table. First, the relative popularity of the various HTTP session headers varies greatly among spam and legitimate Web pages. For example, 60% of the spam instances in the Webb Spam Corpus possess an "X-Powered-By" header, whereas only 29.1% of the legitimate instances in the WebBase data contain that header. This

disparity is even larger for the "Link" header, which is found in 40.9% of the spam instances and only 0.02% of the legitimate instances. The table also shows a similar phenomenon for the spam and legitimate instances in the WEBSPAM-UK2006 corpus.

The other important observation from Table 25 is that distinct distributions exist for the HTTP session header values of spam and legitimate Web pages. For example, of the 59,023 spam instances in the Webb Spam Corpus that possess a "P3P" header value, only 34.5% of them have a value that is also possessed by at least one legitimate instance in the WebBase data (i.e., 65.5% of those spam instances are uniquely identified by their "P3P" header value). Similarly, 17.8% of all spam instances in the Webb Spam Corpus are uniquely identified by their "Content-Type" header value, and 24.8% of the spam instances with a "Server" header value are uniquely identified by those values. Similar distinctions also exist for the HTTP session header values of spam and legitimate Web pages in the WEBSPAM-UK2006 corpus.

To derive classification features from HTTP session headers, we created three representations (phrases, n-grams, and tokens) for each unique header value. First, we stored the header value as an uninterrupted phrase. Then, we tokenized the header value using whitespace and punctuation characters as delimiters. Next, we stored the resulting tokens, along with the unique 2-grams and 3-grams. Finally, we prepended the header name to each of our generated feature values to disambiguate the values for distinct headers. For example, we prepended all "Server" values with "server_" to avoid potential collisions with the values of other headers. To illustrate our feature generation process, Table 26 shows the features that were derived from the sample "Server" header shown in Table 25 ("apache/2.0.52 (fedora)").

10.3.4 Feature Selection

Text data is often characterized as having an extremely high dimensionality due to the vast number of features that can occur in the feature vector. Our corpora exhibit a particularly high dimensionality because they contain thousands of unique headers with millions of unique values. Additionally, we expand the feature space even further with hosting IP

Table 26: Feature representations.

Representation	Feature
Phrase	server_apache/2.0.52 (fedora)
N-grams	server_apache/2 0 52 server_0 52 fedora server_apache/2 0 server_0 52 server_52 fedora
Tokens	server_apache/2 server_0 server_52 server_fedora

addresses and the three feature representations (i.e., phrases, n-grams, and tokens) that we apply to each HTTP header value.

Many of these features are critical for establishing accurate class boundaries and creating effective classifiers. However, not all of these features are relevant to the distinction between spam and legitimate instances. Irrelevant features actually introduce noise into the classification process, wasting valuable computational and storage resources during the training phase and all subsequent uses of the trained classifier. To alleviate the problems associated with high dimensionality, we select a smaller number of the "best" features from our vast feature space – a process known as dimensionality reduction (or feature selection). In our experiments, we use a well-known information theoretic measure called Information Gain [57, 155] to accomplish this feature selection process. Information Gain is defined as follows:

$$IG(f_i, c_j) = \sum_{c \in \{c_j, \overline{c_j}\}} \sum_{f \in \{f_i, \overline{f_i}\}} p(f, c) \cdot log \frac{p(f,c)}{p(f) \cdot p(c)},$$

where f_i is a feature in the feature vector, $p(f)$ is the probability that f occurs in the training set, c_j is one of the classes (i.e., spam or legitimate), $p(c)$ is the probability that c occurs in the training set, and $p(f, c)$ is the joint probability that f and c occur in the training set.

Intuitively, Information Gain quantifies how much the knowledge of feature f_i helps to

correctly identify class c_i. Thus, if feature f_0 has a higher Information Gain value than feature f_1, we say that f_0 has more predictive power than f_1. For our experiments, we calculate the Information Gain for each feature in the feature space of a given collection of corpora. Then, a user-specified number n of tokens with the highest Information Gain scores are selected (or retained) and used to train the classifiers.

10.3.5 Classifiers

Unlike previous Web spam classification research [15, 28, 43, 115], which has relied almost exclusively on decision trees (e.g., C4.5), we performed our HTTP session classification experiments with a variety of classification algorithms. Specifically, we performed an extensive evaluation with 40 classifiers that are implemented in the Weka toolkit [152]. These classifiers include decision trees (e.g., C4.5, random forest, etc.), rule generators (e.g., RIPPER, PART, etc.), boosting algorithms (e.g., AdaBoost, LogitBoost, etc.), logistic regression, radial basis function (RBF) networks, HyperPipes, multilayer perceptrons, support vector machines (SVM), and naïve bayes. The algorithmic details of these classifiers are beyond the scope of this chapter. Additional information about the classifiers is available in most standard machine learning texts (e.g., [110]).

10.4 Webb Spam and WebBase Results

In this section, we present our experimental HTTP session classification results using spam instances from the Webb Spam Corpus and legitimate instances from our WebBase sample. The Webb Spam Corpus is currently the largest publicly available collection of Web spam; consequently, the results presented in this section are a truly representative measure of the effectiveness of our approach.

10.4.1 Classifier Evaluation

Our first experiments explored the impact of feature set size and corpus sample size on the effectiveness of our classifiers. As part of these experiments, we varied the feature set size between 100 and 10,000 (incrementing the variations on a log scale), and we varied the corpus sample size across the same range with the same variations. As a result, we evaluated

Table 27: Top 10 features for Webb Spam and WebBase.

Rank	Feature
1	accept-ranges_bytes
2	x-powered-by_php/4 3
3	x-powered-by_php/4
4	content-type_text/html; charset=utf-8
5	content-type_text/html; charset=iso-8859-1
6	expires_00 00 gmt
7	64.225.154.135
8	server_fedora
9	pragma_no-cache
10	p3p_cp=

close to 400 unique feature set size and corpus sample size combinations. After performing this evaluation, we found that the majority of our classifiers consistently exhibited their best performance with 5,000 retained features (the 10 most effective features for these corpora are summarized in Table 27) and a corpus sample consisting of 10,000 data instances. Therefore, those settings were used for the remainder of our experiments involving the Webb Spam Corpus and WebBase data.

Once we identified an appropriate feature set size and corpus sample size, we used those settings to evaluate the performance of the 40 classifiers we described above in Section 10.3.5. This evaluation was performed using a random corpus sample with an equal distribution of spam and legitimate instances (i.e., 5,000 instances of each class). We chose to use an equal class distribution to minimize the class-specific contributions and focus on each classifier's ability to correctly classify instances from each class. We revisit this decision in Section 10.4.2 by investigating the impact of a corpus sample's class distribution on the performance of a classifier.

The performance metrics for the 5 most effective classifiers from our evaluation are presented in Table 28. The table clearly shows that all of our classifiers were quite successful; however, HyperPipes was consistently the best performer. SVM and C4.5 both exhibit a slightly higher TP rate (i.e., spam detection rate) than HyperPipes, but HyperPipes offers

Table 28: Classifier performance results for Webb Spam and WebBase.

Classifier	TP	FP	F-Measure	Accuracy
Content-based [115]	86.2%	1.3%	0.886	92.5%
C4.5	88.5%	4.6%	0.916	91.9%
HyperPipes	88.2%	**0.4%**	**0.935**	**93.9%**
Logistic Regression	88.2%	2.0%	0.927	93.1%
RBFNetwork	87.1%	0.8%	0.927	93.2%
SVM	**89.4%**	2.3%	0.933	93.6%

a substantially lower FP rate than all of the other classifiers. The HyperPipes classifier operates as follows. During the training phase, an n-dimensional parallel-pipe (or HyperPipe) is constructed for each class. Each HyperPipe contains all of the feature values associated with its class, which generates feature boundaries for that class. During the testing phase, each test instance is classified according to the class that most contains that instance (i.e., the class with feature boundaries that overlap the most with the test instance).

To further investigate the performance of HyperPipes (and the other classifiers), we generated a Receiver Operating Characteristics (ROC) graph. ROC graphs display the TP rate on the Y axis and the FP rate on the X axis, depicting the relative tradeoffs between benefits (true positives) and costs (false positives). In the machine learning community, these graphs are used as another method for comparing the performance of classifiers [51]. Figure 51 shows the ROC graph for our classifiers. As the graph shows, the HyperPipes classifier offers one of the best tradeoffs between TP and FP performance. Since users are more concerned with reducing false positives (i.e., legitimate Web pages that are misclassified as spam) than false negatives (i.e., Web spam pages that are misclassified as legitimate), the HyperPipes classifier offers the best overall performance, and we will rely on it for future evaluations.

To help put our results in their proper context, Table 28 also shows the best results from a previously studied content-based Web spam classifier [115]. Since the results from this previous study were obtained using a private data set, we were unable to generate directly comparable results. However, an indirect comparison shows that all of our top 5

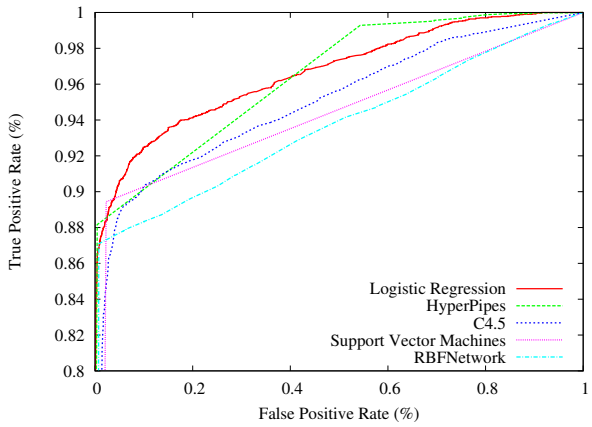


Figure 51: ROC curves for the top 5 classifiers using Webb Spam and WebBase.

classifiers were able to detect a higher percentage of Web spam pages than the content-based classifier (i.e., our classifiers exhibited higher TP rates), and two of our classifiers (HyperPipes and RBFNetwork) generated lower FP rates. Based on these results, we believe that our approach for predicting Web spam with HTTP session information performs as well as (if not better than) content-based Web spam classification efforts.

10.4.2 Class Distribution Evaluation

Based on the empirical growth patterns exhibited by Web spam [74], we expect the percentage of Web spam on the Web to continue growing for the foreseeable future. As a result, any proposed solution for Web spam must be resilient against a constantly evolving distribution of spam and legitimate Web pages. With this in mind, we performed an experiment to evaluate the effect of a corpus sample's class distribution on the performance of a HyperPipes classifier. First, we created 9 corpus samples, each with a different class distribution. The percentage of spam instances in each sample was varied from 10% to 90%, in increments of 10%. Then, for each sample, we selected the 5,000 most informative features and evaluated a HyperPipes classifier using stratified tenfold cross-validation. The results of this evaluation are summarized in Table 29. As the table shows, the classifier's performance declines as the sample's spam percentage increases. However, the classifier's overall performance is incredibly robust. The FP rate remains relatively constant until the sample is composed of 60% spam, and the lowest TP rate, F-Measure, and Accuracy values

Table 29: Class distribution results for Webb Spam and WebBase.

Spam Percentage	TP	FP	F-Measure	Accuracy
10%	87.2%	**0.3%**	0.917	**98.4%**
20%	**91.2%**	**0.3%**	**0.948**	98.0%
30%	89.4%	**0.3%**	0.941	96.6%
40%	89.1%	**0.3%**	0.940	95.5%
50%	88.2%	0.4%	0.935	93.9%
60%	87.1%	0.6%	0.929	92.0%
70%	86.1%	0.8%	0.924	90.0%
80%	85.6%	1.0%	0.921	88.3%
90%	85.2%	3.1%	0.919	86.4%

are all within about 10% of their best values.

10.4.3 Computational Costs

Up to this point, our evaluations have focused primarily on the effectiveness of HTTP session classification. However, another important consideration for our proposed Web spam solution is the computational cost of HTTP session classification. Ultimately, even the most effective classifier is useless if its timing requirements inhibit Web browsers and Web crawlers from performing their intended tasks.

To evaluate the computational requirements of HTTP session classification, we performed timing experiments using the 5 classifiers presented in Section 10.4.1. For each classifier, we computed the training time and per instance classification time necessary to perform a stratified tenfold cross-validation evaluation. Our timing experiments were conducted on a dual processor (Intel Xeon 2.8 GHz) system with 4 GB of memory, and the results for the 3 most efficient classifiers are shown in Table 30. As the table shows, C4.5 and HyperPipes are both extremely efficient when classifying data instances, requiring an average of only 9μs and 101μs, respectively. RBFNetwork is more than two orders of magnitude slower than HyperPipes, requiring an average of 14.32ms to classify each instance. The table also shows that HyperPipes is the most efficient classifier in terms of training time (408.8ms). The training times of C4.5 (5,389.1s) and RBFNetwork (734.99s) are both more than three orders of magnitude larger than the corresponding training time for HyperPipes. However, it is important to note that these training times are only incurred when the

Table 30: Classifier training and per instance classification times using Webb Spam and WebBase.

Classifier	Training Time		Classify Time Per Instance	
	Avg. (s)	σ (s)	Avg. (ms)	σ (ms)
C4.5	5,389.1	3.67	**0.009**	**0.002**
HyperPipes	**0.4088**	**0.03**	0.101	0.008
RBFNetwork	734.99	8.16	14.32	0.033

classifiers are trained (or retrained), which is an infrequent occurrence. Regardless, our HyperPipes classifier provides the most efficient HTTP session classification solution, adding a mere 101μs to each HTTP request operation. This timing requirement will be completely unnoticeable by Web users, and it should also have almost no effect on the operations of Web crawlers.

10.4.4 Resource Savings

One of the primary benefits of our predictive approach is that it identifies Web spam without downloading the contents of Web spam pages. By eliminating these downloads, we generate significant bandwidth and storage space savings for Web browsers and Web crawlers.

To help quantify the resource savings associated with our approach, we calculated size information for our spam instances. Specifically, we determined that the average size of a Web spam response message in the Webb Spam Corpus is about 16 KB (15.4 KB for the message body and 0.6 KB for the headers). Hence, if our approach successfully detected 100% of the Web spam pages in the corpus, we would save about 15.4 KB in bandwidth and storage costs every time we encountered a spam URL. Based on the actual detection rate (88.2%) exhibited by our HyperPipes classifier, we expect to save an average of about 13.6 KB for every encountered spam URL.

10.5 WEBSPAM-UK2006 Results

In Section 10.4, we showcased the effectiveness of our predictive approach to Web spam classification on large, diverse corpora. The Webb Spam Corpus and WebBase data were drawn from a variety of domains (e.g., .com, .net, .info, etc.), and each collection contains

Table 31: Top 10 features for WEBSPAM-UK2006.

Rank	Feature
1	x-powered-by_php/4
2	x-powered-by_php/4 4
3	set-cookie_path=/
4	server_apache/1.3.33 (unix)
5	cache-control_private
6	pics-label_gov uk
7	serveri_microsoft-iis/5.0
8	212.100.249.135
9	212.227.240.69
10	content-type_text/html;charset=iso-8859-1

more than 300,000 pages. In this section, we focus on the smaller, more focused WEBSPAM-UK2006 corpus. This collection has the advantage of being more recent than our other corpora (July 2007 versus December 2005), but it only contains a few thousand pages that were all drawn from the .uk domain. Therefore, the corpus was used primarily to investigate the robustness of our predictive method in such a different setting.

10.5.1 Classifier Evaluation

To evaluate the effectiveness of HTTP session classifiers on the WEBSPAM-UK2006 corpus, we used the same methodology that we described in Section 10.4.1. First, we explored the impact of the feature set size, and again, we found that the majority of the classifiers generated their best results with 5,000 retained features (the 10 most effective features for the WEBSPAM-UK2006 corpus are summarized in Table 31). Due to the limited size of this corpus, we did not evaluate the impact of corpus sample size on the classifiers. Instead, we chose to use a sample size of 1,486 because that is the largest sample size, given the available data, that would allow us to reproduce the class distribution evaluation described in Section 10.4.2. Both of these settings were used for the remainder of our experiments involving the WEBSPAM-UK2006 corpus.

After we determined an appropriate feature set size and corpus sample size, we used a random corpus sample with an equal distribution of spam and legitimate instances (i.e.,

Table 32: Classifier performance results for WEBSPAM-UK2006.

Classifier	TP	FP	F-Measure	Accuracy
Hybrid [28]	88.4%	6.3%	0.763	**91.1%**
C4.5	79.1%	22.9%	0.783	78.1%
HyperPipes	71.9%	**2.2%**	0.826	84.9%
Logistic Regression	86.1%	19.8%	0.837	86.0%
RBFNetwork	**96.1%**	19.9%	**0.890**	88.1%
SVM	84.3%	22.1%	0.817	81.1%

743 instances of each class) to evaluate the performance of the 40 classification algorithms described in Section 10.3.5. Table 32 summarizes the results of the 5 most effective classifiers from this evaluation. As the table illustrates, the TP rates for all of the classifiers (except RBFNetwork) declined compared to their performance in Table 28, and the FP rates for all of the classifiers increased. A similar performance degradation is also evident in the ROC graph shown in Figure 52.

At least two explanations exist for the performance degradation of the classifiers on this corpus. First, our corpus sample size (1,486) for this evaluation was almost an order of magnitude smaller than the corpus sample size used for the evaluation in Section 10.4.1. Unfortunately, this was unavoidable due to the limited size of the WEBSPAM-UK2006 corpus. Second, previous research [28] noted several times that the content of the spam and legitimate pages in this corpus is more similar than in other corpora. Thus, it is not unreasonable to expect the HTTP session information associated with the spam and legitimate pages in this corpus to be equally similar, making it much more difficult to correctly identify the class boundaries.

Table 32 also shows the best results from a previously studied Web spam classifier that was trained on this corpus using both content-based and link-based features (i.e., a hybrid technique) [28]. It is important to note that these previous results were generated with a highly optimized decision tree algorithm that utilized multiple passes of stacked graphical learning. Our classifiers, on the other hand, were built using default settings and without the benefit of meta-learning optimizations, and even without these optimizations, they generated comparable results. In fact, our HyperPipes classifier actually produced a

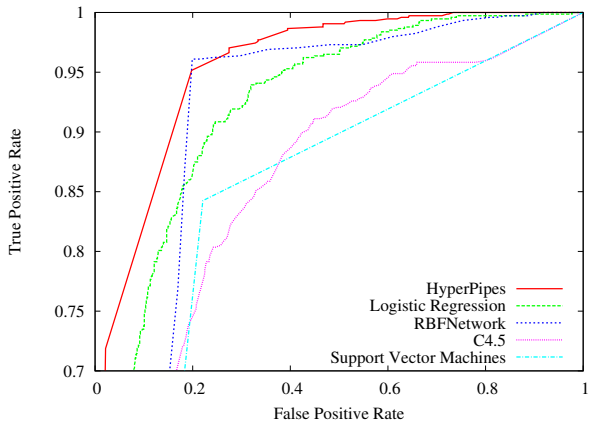


Figure 52: ROC curves for the top 5 classifiers using WEBSPAM-UK2006.

significantly lower FP rate, which prompted us to use it for the remaining evaluations.

10.5.2 Class Distribution Evaluation

As mentioned above in Section 10.4.2, effective Web spam solutions must be able to handle an evolving distribution of spam and legitimate Web pages. Therefore, after we identified HyperPipes as the best classifier for this corpus, we evaluated its resiliency against various class distributions. Using the same approach described in Section 10.4.2, we evaluated the classifier on 9 corpus samples with class distributions varying from 10% spam to 90% spam, in increments of 10%. The results of this evaluation are shown in Table 33. The table shows somewhat erratic TP rates, FP rates, and F-Measure values as the percentage of spam increases. Unlike Table 29, which shows a slow decline for these metrics as the spam percentage increases, the results in Table 33 exhibit distinct patterns of increasing and decreasing performance. When the spam percentage is between 30% and 70%, the classifier's performance is relatively consistent, exhibiting TP rates between 62.7% and 73.1% and FP rates between 1.0% and 2.2%. However, the classifier's performance varies widely for the most extreme class distributions (i.e., spam percentages of 10%, 20%, 80%, and 90%).

The most logical explanation for the classifier's performance variations is the small size of this corpus. When the spam percentage was 10%, the sample only contained 149 spam instances. This small spam sample made it extremely difficult for the classifier to learn

Table 33: Class distribution results for WEBSPAM-UK2006.

Spam Percentage	TP	FP	F-Measure	Accuracy
10%	36.2%	1.5%	0.484	**92.3%**
20%	59.9%	**0.9%**	0.733	91.3%
30%	71.5%	1.8%	0.814	90.2%
40%	**73.1%**	1.5%	**0.834**	88.4%
50%	71.9%	2.2%	0.826	84.9%
60%	62.7%	1.0%	0.767	77.2%
70%	70.8%	2.0%	0.825	78.9%
80%	65.4%	3.7%	0.787	71.6%
90%	67.2%	10.1%	0.798	69.4%

distinguishing spam features, and the classifier's TP rate decreased dramatically. Similarly, when the spam percentage was 90%, the sample only contained 149 legitimate instances. This small number of legitimate instances made it quite challenging for the classifier to learn distinguishing legitimate features; consequently, the classifier's FP rate increased. If the WEBSPAM-UK2006 corpus was larger and we could derive a sample size comparable to the one used in Section 10.4.2, we believe these performance fluctuations would be reduced.

10.5.3 Computational Costs

To evaluate the computational costs of HTTP session classification on the WEBSPAM-UK2006 corpus, we performed timing experiments with the 5 classifiers presented in Section 10.5.1. Using the same approach described in Section 10.4.3, we computed the training time and per instance classification time necessary for each classifier to perform a stratified tenfold cross-validation evaluation. Table 34 shows the results for the 3 most efficient classifiers. The relative performance of the classifiers shown in the table is the same as in Table 30. However, the training times in Table 34 are all much lower because the corpus sample size for this evaluation was almost an order of magnitude smaller. Additionally, the per classification times for C4.5 and HyperPipes actually increased slightly, whereas the corresponding time for RBFNetwork decreased. We believe these slight variations are another artifact of the small sample size, which is reaffirmed by the higher standard deviation values shown in Table 34. As a result, we believe the timings obtained on the larger corpora in Section 10.4.3 are more indicative of our approach's efficiency.

Table 34: Classifier training and per instance classification times using WEBSPAM-UK2006.

Classifier	Training Time		Classify Time Per Instance	
	Avg. (s)	σ (s)	Avg. (ms)	σ (ms)
C4.5	134.21	1.95	**0.010**	**0.007**
HyperPipes	**0.1480**	**0.01**	0.232	0.010
RBFNetwork	51.556	0.42	8.332	0.233

10.5.4 Resource Savings

By avoiding the costly downloads associated with Web spam pages, our approach saves valuable bandwidth and storage resources for Web browsers and Web crawlers. To help quantify the magnitude of these savings, we investigated the size characteristics of the Web spam pages in the WEBSPAM-UK2006 corpus. The average size of a Web spam response message in the corpus is about 24 KB (23.3 KB for the message body and 0.7 KB for the headers). Hence, if our approach successfully detected 100% of the Web spam pages, we would save about 23.3 KB in bandwidth and storage costs every time we encountered a spam URL. Based on the actual detection rate (71.9%) exhibited by our HyperPipes classifier, we expect to save an average of about 16.8 KB for every encountered spam URL.

10.6 Summary

In this chapter, we have presented a predictive approach to Web spam classification that relies exclusively on HTTP session information (i.e., hosting IP addresses and HTTP session headers), and we used this predictive approach to build an HTTP session classifier. Using a corpus of almost 350,000 Web spam instances and almost 400,000 legitimate instances, our HTTP session classifier effectively detected 88.2% of the Web spam pages with a false positive rate of only 0.4%. Additionally, by incorporating this classifier into the HTTP retrieval process, our approach was capable of saving an average of 15.4 KB of bandwidth and storage resources for every successfully identified Web spam page, while only adding an average of 101μs to each HTTP retrieval operation.

CHAPTER XI

EXPLORING THE DARK SIDE OF SOCIAL ENVIRONMENTS

Over the past few years, social networking communities have experienced unprecedented growth. Communities such as MySpace and Facebook are connecting people in a variety of new and exciting ways, and as a result, individuals are attaching an increasing amount of value to their online personas. Unfortunately, the rising importance and prominence of these communities have also made them prime targets for attack by malicious entities.

We observe two distinct attack classes that threaten social networking communities and the privacy of their users. First, traditional attacks that have plagued Internet users for many years (e.g., malware propagation, spam, and phishing) have been adapted to take advantage of the unique properties of online communities. These traditional attacks are thriving in their new environment, and the severity of these attacks promises to grow as attackers become more sophisticated. The second attack class consists of new attacks that have emerged from the very fabric of these communities. One prominent example is the use of deceptive profiles (e.g., rogue advertising profiles and impersonating profiles) that are becoming more widespread, difficult to detect, and extremely costly to legitimate community members.

In this chapter, we provide detailed descriptions for each of these attack classes, and we show that the continued success of social networking communities is contingent upon their ability to mitigate the risks associated with these attacks. For concreteness, we describe attacks that are observable in MySpace, which is the most popular social networking community in terms of unique visitors (more than 65 million in February 2008) [59], total traffic (4.29% of all U.S. Internet visits in February 2008) [83], and user base (more than 110 million active accounts as of February 2008) [143]. Our observations show the practical importance of these attacks and their impact on millions of users. Additionally, since

other social networking communities are both functionally and structurally similar to MySpace, the attacks we describe can be easily adapted to most (if not all) social networking communities.

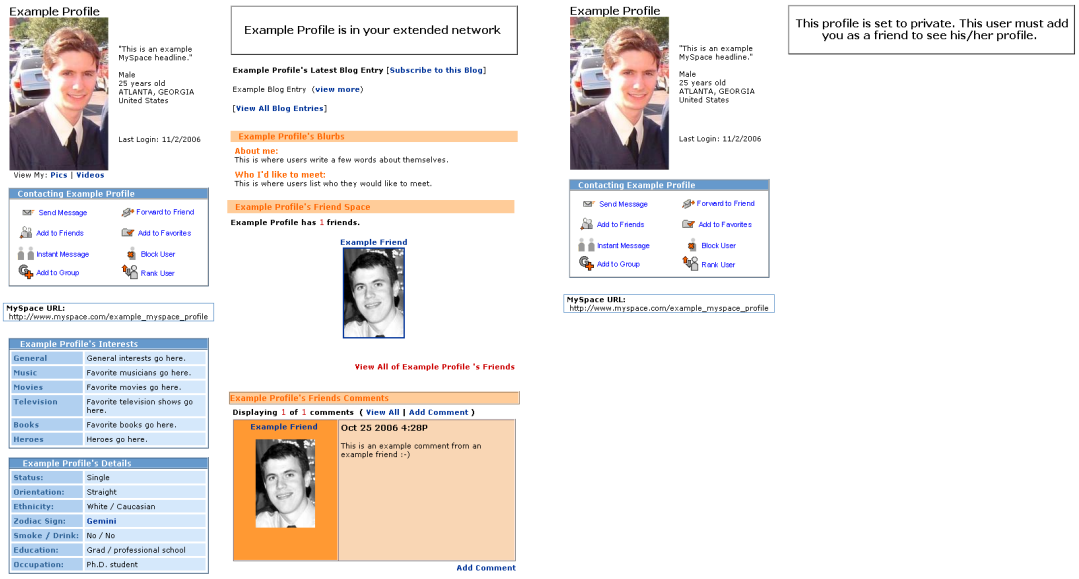
The rest of the chapter is organized as follows. Section 11.1 provides background information about social networking communities. In Section 11.2, we describe traditional attacks that have adapted to target social networking environments, including malware propagation, spam, and phishing. In Section 11.3, we present new attacks that specifically target social networking communities by utilizing deceptive profiles such as rogue advertising profiles and impersonating profiles. Section 11.4 summarizes our findings.

11.1 Background on Social Networking Communities

Social networking communities provide an online platform for people to manage existing relationships, form new ones, and engage in a variety of social interactions. Typically, a user's online presence in these communities is represented by profile, which is a user-controlled Web page that contains a picture of the user along with various pieces of personal information. Profiles connect to other profiles through explicitly declared friend relationships and numerous messaging mechanisms.

An example of a MySpace profile is shown in Figure 53(a). Some of a profile's personal information is mandatory (e.g., the user's name, age, gender, location, etc.), and some of it is optional (e.g., the user's interests, relationship status, occupation, etc.). To facilitate self-expression, each profile also has free text "About me" and "Who I'd like to meet" sections, and users are allowed to embed various objects in their profiles such as pictures, audio clips, and videos. In addition to personal information and embedded content, a user's profile also contains a list of links to the profiles of that user's friends. These friend links are bidirectional because a link is established only after both parties acknowledge the friendship. To initiate a friendship, a user sends a friend request to another user. If the other user accepts this request, the friendship is established, and a friend link is added to both users' profiles.

Aside from friend requests, MySpace provides a number of other communication facilities



(a) Public Profile (b) Private Profile

Figure 53: Example MySpace profiles. In (a), the profile is publicly accessible. In (b), the profile is private.

that enable users to communicate with each other within the community. These facilities include messaging, bulletin, commenting, blogging, and instant messaging (IM) systems. The messaging system allows users to exchange intra-community email messages with any other user (i.e., both friends and strangers). The bulletin system is essentially an exclusive bulletin board that only a user's friends can view, enabling users to communicate with all of their friends at once. Users can post comments on their friends' profiles using the commenting system, and the blogging system allows users to maintain blogs on their profiles, which other users can read and comment on. Finally, the IM system provides users with a mechanism to send intra-community instant messages to any other user.

Due to the wealth of private information that is accessible on user profiles and the various means of communication that are available, MySpace provides mechanisms to protect the privacy of its users. First and foremost, users have the ability to choose between making their profiles publicly viewable (the default option) or private. If a user's profile is designated as private, only the user's friends are allowed to view the profile's detailed personal information (e.g., the user's interests, blog entries, comments, etc.). However, as

Figure 53(b) shows, a private profile still reveals the user's name, picture, headline, gender, age, location, and last login date. MySpace also provides a few finer-grained privacy mechanisms. Users can control who is allowed to IM them (everyone, only friends, or no one) and who is allowed to leave blog comments (everyone or only friends). Users can also maintain a block list to prevent specific users from contacting them at all. Unfortunately, as we will see in the following sections, these privacy mechanisms have been unable to prevent a number of attacks.

11.2 Traditional Attacks Targeting Social Networking Communities

Since social networking communities include communication facilities that are fundamentally similar to traditional means (i.e., email, blogs, instant messaging, etc.), many of the attacks that are effective against those traditional communication media have been adapted to exploit social network communications. Due to the massive size of many social networking communities, their tightly connected nature, and their relatively naïve user bases, these communities are target rich environments for attackers. In this section, we describe three of the adapted attacks that have been observed in MySpace: malware propagation, spam, and phishing.

11.2.1 Malware Propagation

Malware creators aim to spread their malicious content to as many victims as possible. Since MySpace is the most popular community on the Web, it has become a prime target for malware propagation. In fact, over the past couple of years, MySpace was attacked by at least one instance of each of the following malware categories: worms, spyware, and adware. For the remainder of this section, we will detail the most interesting occurrences of these attacks, and we will explain the threats they pose to social networking communities.

11.2.1.1 Worms

The most successful example of rapid worm propagation in a social networking community occurred in MySpace on October 4, 2005. The worm was called the "Samy worm," and it generated more than a million friend requests for its creator (Samy) over the course of a

single day. The basic operation of this worm was quite simple but extremely clever [130]. First, Samy wrote the worm using Javascript, and then, he embedded it in his MySpace profile. MySpace disallows users from adding scripts to their profiles by filtering scripting tags and removing specific strings (e.g., "javascript"). To evade these filters, Samy exploited the behavior of popular Web browsers such as Internet Explorer. Specifically, he hid the code inside a Cascading Style Sheet (CSS) tag and obfuscated the strings that MySpace would filter (e.g., "javascript" became "java\nscript"). Thus, even though MySpace had security mechanisms in place, the lax security of certain Web browsers allowed the obfuscated code to execute successfully.

When a MySpace user accessed Samy's profile, the embedded Javascript code sent Samy a friend request on that user's behalf. The code also embedded itself in the user's profile (in the same manner that it was embedded in Samy's profile). Consequently, when other MySpace users visited the newly infected profile, the code would send Samy friend requests from those users and propagate itself to their profiles. As Samy described it, "If 5 people viewed my profile, that's 5 new friends. If 5 people viewed each of their profiles, that's 25 more new friends."

Fortunately, this worm was relatively harmless for its infected users because it only affected their MySpace profiles and not their actual machines. However, the same cannot be said for MySpace. Due to the worm's viral growth pattern, the MySpace administrators were forced to temporarily shut down the site to stop the worm's propagation and remove the worm's code from the infected users' profiles. The service outage was relatively brief, but this incident clearly illustrates the potential damage that social networking worms are capable of inflicting. Specifically, this worm teaches two very important lessons. First, the success of the worm's obfuscated code clearly highlights the importance of Web site security as well as Web browser security. A delicate balance exists between functionality and security, and this balance must be respected when designing and developing online communities. Second, the worm's propagation speed showcases how quickly the entire community could become infected. In this case, more than a million users were affected in less than 24 hours.

A much more dangerous social networking worm appeared on MySpace in the middle of July 2006. Similar to the Samy worm, this worm propagated itself through the profiles of unsuspecting MySpace users. However, unlike the Samy worm, this worm's code was hidden inside a malformed Shockwave Flash (.swf) file that directly exploited a vulnerability on users' machines. Additionally, instead of generating innocuous friend requests, this worm actually redirected users' Web browsers to a politically charged blog posting.

The manner in which this worm spread is as follows. First, the worm's creator generated a malformed .swf file and embedded it in a MySpace profile. This .swf file exploited a critical vulnerability in Macromedia Flash Player v8.0.24.0 and earlier versions, which allows an attacker to execute code on an affected machine. When a MySpace user with a vulnerable Macromedia Flash Player accessed the profile, the code in the .swf file automatically redirected the user's browser to a blog posting that contained political propaganda. Finally, the code propagated itself by embedding a copy of the .swf file in the infected user's profile.

This worm was also relatively harmless; however, it was far more troubling than the Samy worm because it actually exploited a vulnerability on users' computers, which allowed it to execute code. Fortunately, the worm's creator was only interested in spreading political ideals because nothing prevented the worm from installing any number of malicious utilities on its victims' machines. For example, the worm could have installed spyware or adware, and it could have even zombified the infected machine (i.e., compromise the computer and enlist it in a botnet [62]). Combine these frightening, yet completely realistic, scenarios with the viral propagation patterns of these worms, and it becomes immediately obvious how devastating worms could be in a social networking environment and why they must be prevented.

11.2.1.2 Spyware

Although spyware has yet to be distributed within the payload of a social networking worm, it has already made a few appearances in social networking communities. The most prominent example of spyware appearing in MySpace occurred towards the beginning of

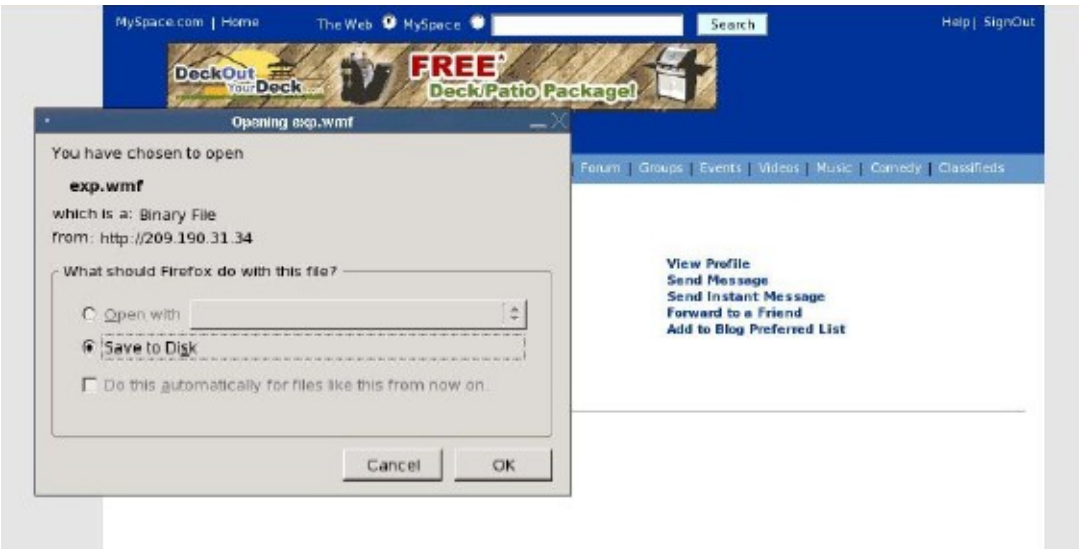


Figure 54: Embedded .wmf image within a deckoutyourdeck.com advertisement.

July 2006. At that time, an advertisement for deckoutyourdeck.com, which contained a malformed Windows Metafile (.wmf) image, was inserted into one of the ad networks that MySpace uses. This malformed .wmf image exploited a critical vulnerability in the Graphics Rendering Engine of Windows, which allows remote code execution.

MySpace displays an ad banner, which is randomly selected from MySpace's supplying ad networks, at the top of every profile. When a user accessed a profile that displayed the deckoutyourdeck.com ad, the user was prompted to download the embedded .wmf image. Figure 54 shows a screenshot of this prompted download. If the user was running an unpatched version of Windows, the .wmf file installed an assortment of programs, including known spyware utilities such as PurityScan [98]. These spyware utilities pose a serious threat to the user because they track the user's Web browsing behaviors, and they install various other third-party applications without the user's consent.

Unfortunately, despite the fact that Microsoft released a patch for this vulnerability more than six months before the ad appeared on MySpace, over a million users were affected. Thus, this incident reveals two very important points. First, many users have a false sense of security in these communities. One of the most basic secure browsing principles is never to download questionable content, yet more than a million users gladly accepted this suspicious (and completely unsolicited) download request. Second, an alarming number of

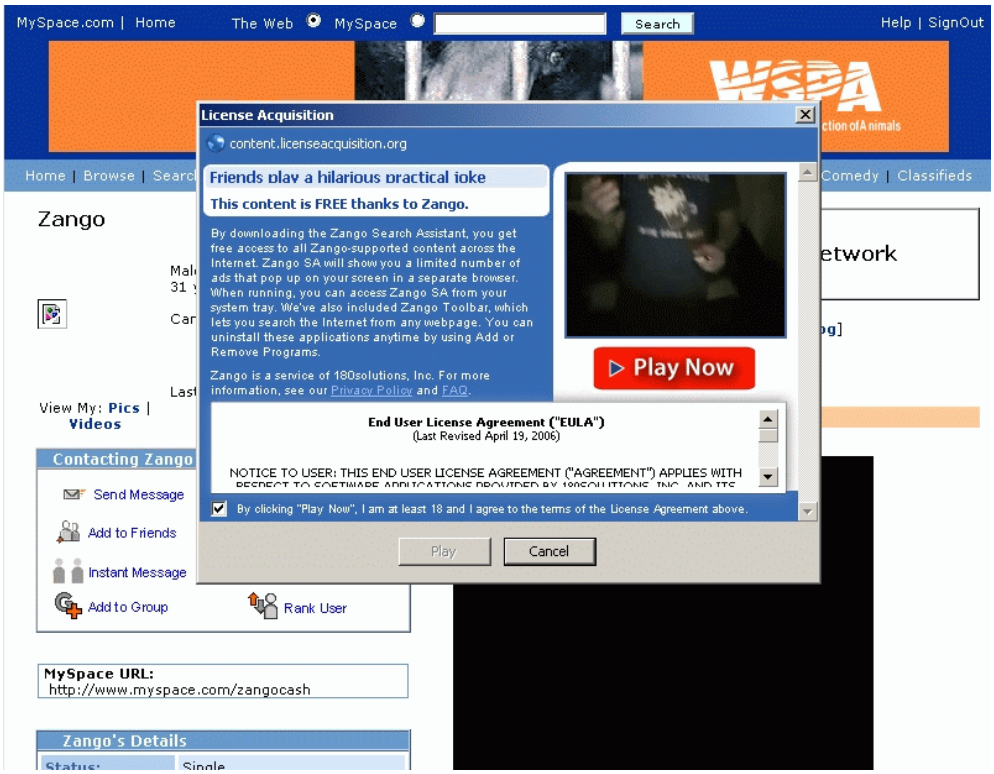


Figure 55: Deceptive Zango popup license agreement.

users are not vigilant about protecting themselves against security threats. This incident proves that at least a million users neglected to install security patches for more than six months. Therefore, social networking communities must assume that their users are completely vulnerable and take every precaution necessary to protect them from malicious content.

11.2.1.3 Adware

An interesting instance of adware appearing in MySpace occurred around the same time as the previous spyware example. A security researcher was browsing MySpace profiles and found two that were named after a known adware company called "Zango" (formerly known as "180solutions"). Both profiles were created to deceive users into downloading and installing the Zango Search Assistant and Toolbar (two known adware programs) [20]. The first profile claimed that the programs could "protect kids from predators," and the second profile was even more deceptive.

When a user accessed the second Zango profile, a popup with a license agreement immediately launched, asking the user to accept the license in order to play a video file. This license popup is shown in Figure 55. If the user accepted the license by clicking the "Play Now" button, a video file began to play, but secretly, the Zango Search Assistant and Toolbar were also installed on the user's system. As Figure 55 illustrates, this popup contained a number of deceptive elements. First, the popup did not explicitly indicate that an installation was taking place. The text in the top-left corner of the popup mentions the Zango Search Assistant and Toolbar, but it does not tell the user that they will be installed. Additionally, none of the popup's buttons mention an installation. Instead, they are entitled "Play Now" and "Play," which implies that the only action will be the play-back of a video file. Finally, the most prominent features of the popup are the video's preview window in the top-right corner and the "Play Now" button. The actual license agreement and its preselected checkbox are positioned at the bottom, where they are likely to be overlooked. Consequently, many users played the video before they noticed the license agreement, and the adware was successfully installed on their machines.

According to Zango, both profiles were created by a Zango developer that was acting against the company's policy of not targeting MySpace. As a result, the deceptive video clip was removed, and the company released a public apology. However, this occurrence clearly illustrates the potential for abuse by adware companies in social networking communities. Additionally, the success of this deception further illustrates the naïvety of users and the need for these communities to provide effective security mechanisms that protect against malicious content.

11.2.2 Spam

In addition to malware propagation, another type of attack on traditional communication media (e.g., email, blogs, instant messaging, etc.) is spam. Since MySpace provides similar communication facilities, they are also susceptible to spamming abuse. In fact, almost all spamming activities that occur outside the community can be recreated inside the community. To aggravate the problem, spammers can also use the profile information posted by

users to target them outside the social networking community. Although MySpace's Terms of Use Agreement prohibits users from providing contact information such as email addresses and URLs on their profiles, many users still do so at their own peril. Consequently, spammers can use that information to spam those users using traditional techniques.

Spamming relies on the open nature of communications; thus, the vulnerabilities of MySpace's communication facilities are proportionate to the openness of their access. On the conservative side, the commenting and bulletin systems are the most spam resistant because they can only be used by a user's friends. The blogging, IM, and friend request systems also provide the option of disallowing non-friends from using them to contact a user, but this protection is not enabled by default. Thus, these three systems are more susceptible to spamming because any user can use them to contact other users that have not enabled this protection. The messaging system is the most vulnerable to abuse because it does not provide an effective spam prevention mechanism. It allows users to report a message as being spam, but those users have no assurances that immediate action will be taken. Therefore, users may receive a number of spam messages before MySpace administrators eliminate the offending spammer from the system. MySpace also provides a general block listing mechanism that allows users to completely prevent all communication from specific users. However, this feature is ineffective due to the ease with which users can create new MySpace profiles (i.e., if spammers have been blocked, they can create new profiles and continue their spamming activities undeterred).

Spammers abuse these communication systems for the same reason they abuse traditional communication facilities: promotion. Spammers want to expose their products, Web sites, and viewpoints to as many individuals as possible, and spamming provides them with a technique to accomplish that goal. For social networking communities, this spamming activity represents a huge problem for a couple of reasons. First, spam content wastes a significant amount of resources, including storage space, bandwidth, and users' time. This last wasted resource leads us to the second major consequence of social network spam. When users waste time dealing with spam, it has a negative effect on their overall experience with these communities because it prevents them from participating in their desired activities.

For example, when users are forced to sift through an inbox full of annoying spam messages, they are unable to send and receive messages. Similarly, when users are burdened with removing an endless stream of spam comments on their blogs, they are unable to post new content. Consequently, if users become overwhelmed with spam, they will have no incentive to continue interacting with these communities, and the communities will be forced to shut down.

In addition to spamming the social networking community, spammers are also able to use the information on user profiles to more effectively spam users outside the community. By mining email addresses, IM screen names, and other contact information from these profiles, spammers can spam users using traditional techniques (e.g., email spam, spim, blog comment spam, etc.). Additionally, spammers can use the personal information on these profiles to construct user-specific spam content that is more relevant to the user, making it more likely to be read. For example, if a user's profile contains a great deal of sports-related information, a spammer can leverage this information to create a sports-oriented spam message for that user. Since the message's content matches the user's interests, the user is much more likely to read it, and as a result, the spam's sales pitch is more likely to be successful. To make their messages even more deceptive, spammers can also masquerade as users' friends. A user's profile contains a list of that user's friends; thus, spammers can use this information to make their messages appear as if they were sent by those friends. Since a user is much more likely to read and trust a message from a friend, the spammer has a higher probability of success with these disguised messages [89].

11.2.3 Phishing

In a traditional phishing attack, a phisher seeks to obtain a targeted user's sensitive information through means of deception. Historically, phishers have been interested in credit card information, banking information, and login information for various Web sites (e.g., eBay, PayPal, etc.). As social networking communities have become more popular, complex networks of friends have been established within them. Consequently, social networking communities have become a prime target for phishers because they are portals to a large

number of potential victims.

A MySpace phishing attack utilizes the same deceptive techniques that are employed in traditional phishing attacks. First, a user is presented with a seemingly legitimate URL (called a *phishing URL*) that appears to be affiliated with MySpace. This phishing URL is typically propagated in two distinct manners: the communication systems provided by MySpace and traditional communication channels (e.g., email, blogs, instant messaging, etc.). For example, in May 2006, a phisher used the MySpace spamming techniques described above to send a phishing message to various MySpace users. This message had "CHECK OUT these old school pictures..." as its subject and a phishing URL in its body [144]. Upon accessing one of these phishing URLs, the user is directed to a fraudulent Web page that appears identical to the authentic MySpace login page. When the user enters the necessary MySpace login information, the fraudulent page stores that information and uses it to redirect the user to the authentic MySpace community. Thus, the user is completely unaware of the attack, while the phisher successfully obtains the user's sensitive login information.

Phishing attacks represent a serious threat to MySpace users for three reasons. First, victimized users often lose control of their profiles. In many cases, phishers will immediately change the login information of profiles they have compromised, locking victims out of their own profiles. Since users spend a great deal of time and energy building new friendships, writing blogs, and customizing their profiles, it is somewhat traumatic when they lose control of their creations. Even in cases where a compromised profile's login information remains the same, the phisher's activities severely damage that profile's credibility. For example, if a phisher compromises a user's profile and begins spamming that user's friends, those friends will eventually distrust the compromised profile and remove it from their lists of friends. The second reason these attacks are dangerous is due to the viral propagation patterns that are possible in social networking communities. As shown with the malware examples above, one compromised user can quickly escalate into a network full of compromised users. Specifically, once a single user's login information is compromised, the phisher can use that user's profile to propagate the attack to the user's friends. In our spam discussion, we mentioned a few MySpace communication systems that are somewhat spam resistant. However, the spam

resistance of those systems assumes that a user's friends can be trusted (i.e., they are not spammers, phishers, etc.). Thus, if a user's profile becomes compromised, that user's friends are immediately at risk because the phisher can contact them under the guise of a profile they trust. As a result, a compromised user's friends are highly likely to become phishing victims (i.e., access the phishing URL), and by the time they realize they should distrust the compromised profile, it will be too late. The final threat posed by MySpace phishing attacks relies on the knowledge that many computer users reuse the same login information at many different sites [126]. Thus, if a user's MySpace login information is compromised, that user's login information at those other sites is automatically compromised as well.

In addition to launching phishing attacks that specifically target social networking communities, phishers can also use the private information found in these communities to make their traditional phishing attacks more effective. As previously mentioned, many MySpace users include various pieces of personal information on their profiles (e.g., location, interests, occupation, etc.). Phishers can easily use this personal information to construct user-specific messages that a potential victim would be much more likely to read and trust. For example, a phisher could send the potential victim a fraudulent eBay email that contains auction information for products the victim would be interested in buying. Each of these products could be selected using the user's interests and associated with one or more phishing URLs. Since this email message contains user-specific content, the victim is more likely to read it and access one of its phishing URLs. As a result, this user-specific attack has a higher probability of success than a generic phishing attack.

11.3 New Attacks Against Social Networking Communities

Initially, social networking communities were composed of ordinary people – typically kids and young adults that created profiles to interact with their friends. However, as these communities began to expand, new parties became interested because the communities evolved into portals for connecting and interacting with these ordinary people. The first wave of new participants were entertainers (e.g., musicians, comedians, and actors) that created profiles to communicate with their current fans and reach out to potential new ones.

Not long after that, companies began partnering with communities to create advertising profiles (i.e., profiles that advertise the company and its products). Eventually, other public figures such as politicians became involved, creating profiles to promote their campaigns, solicit volunteers, and request donations. Unfortunately, the explosive growth of these communities and the conflicting interests of their various participants have generated a range of new attacks. In this section, we describe attacks that utilize two types of malicious social networking profiles: rogue advertising profiles and impersonating profiles.

11.3.1 Rogue Advertising Profiles

According to analysts at eMarketer, companies currently spend around $280 million on social network advertising in the U.S., and by 2010, that figure is expected to grow to $1.9 billion [94]. Thus, it is not surprising that MySpace has partnerships with numerous companies, which allow those companies to advertise legitimately in the social networking community. For example, both Burger King and Wendy's have embraced the potential of social network advertising by creating advertising profiles on MySpace. Burger King uses the King, a marketing character that appears in the company's commercials, to promote its MySpace profile at `http://www.myspace.com/burgerking`. Similarly, Wendy's uses a small hamburger patty named Smart to promote its MySpace profile at `http://www.myspace.com/wendysquare`. However, this new advertising craze is not exclusive to fast food restaurants. MySpace profiles also exist for television shows (e.g., "It's Always Sunny in Philadelphia"), commercial products (e.g., Herbal Essences shampoo), and movies (e.g., "Talladega Nights").

Although MySpace's Terms of Use Agreement strictly prohibits commercial use of the community without prior approval, a number of companies violate this policy by creating advertising profiles without MySpace's consent. We refer to these profiles as *rogue advertising profiles*, and they appear in many forms with varying degrees of deception. The most innocuous group of rogue advertising profiles is created by small Web site operators that are merely trying to generate Web traffic for their sites. These profiles are relatively harmless because they simply include URLs for these small sites, and they are openly affiliated with

a company (i.e., they do not attempt to deceive users into believing the profiles belong to an ordinary individual). However, these profiles are still unacceptable because they pollute the community with unwanted and unauthorized advertising.

A much more deceptive group of rogue advertising profiles is generated by nefarious companies such as gambling and pornographic Web sites. Unlike legitimate advertising profiles, which are clearly identifiable as a marketing device, these deceptive rogue advertising profiles appear as though they are maintained by an individual (and not a company). As a result, naïve users can easily mistake these rogue advertising profiles as ordinary MySpace profiles that were created by ordinary MySpace users. The deceptive profile construction process proceeds as follows. First, a fraudulent MySpace profile is created using a picture of an attractive female. This profile also contains a provocative description in its "About me" section that visitors assume was written by the pictured female. Conveniently, this description usually includes at least one reference to the URL of a nefarious Web site. If the profile is particularly deceptive, the description includes an instant messenger screen name instead of a URL. The inclusion of a screen name is especially manipulative because most users assume that a conversation over IM can only be accomplished by a real person. However, the screen names found on these rogue advertising profiles are almost always attached to an IM bot (i.e., a computer program that emulates a real conversation) that attempts to direct its victims to the URL of a nefarious Web site. Once the profile is completed, its creator sends friend requests to males near a specific geographic location, which the profile is also associated with (i.e., the pictured female claims to live there). When the males receive these requests, a number of them accept and visit the profile, making the deception a success.

Figure 56 shows four examples of deceptive rogue advertising profiles, which we slightly modified to hide objectionable content. By analyzing these examples, a number of interesting observations emerge. First, the profiles are extremely similar. Figures 56(a) and 56(b) share the exact same "About me" text, and Figures 56(c) and 56(d) share the same "About me" text and profile picture. Thus, these two pairs of profiles were probably created by the same individuals. The next observation addresses the content of each profile's "About me" text. All four of these profiles are particularly deceptive because they all provide instant

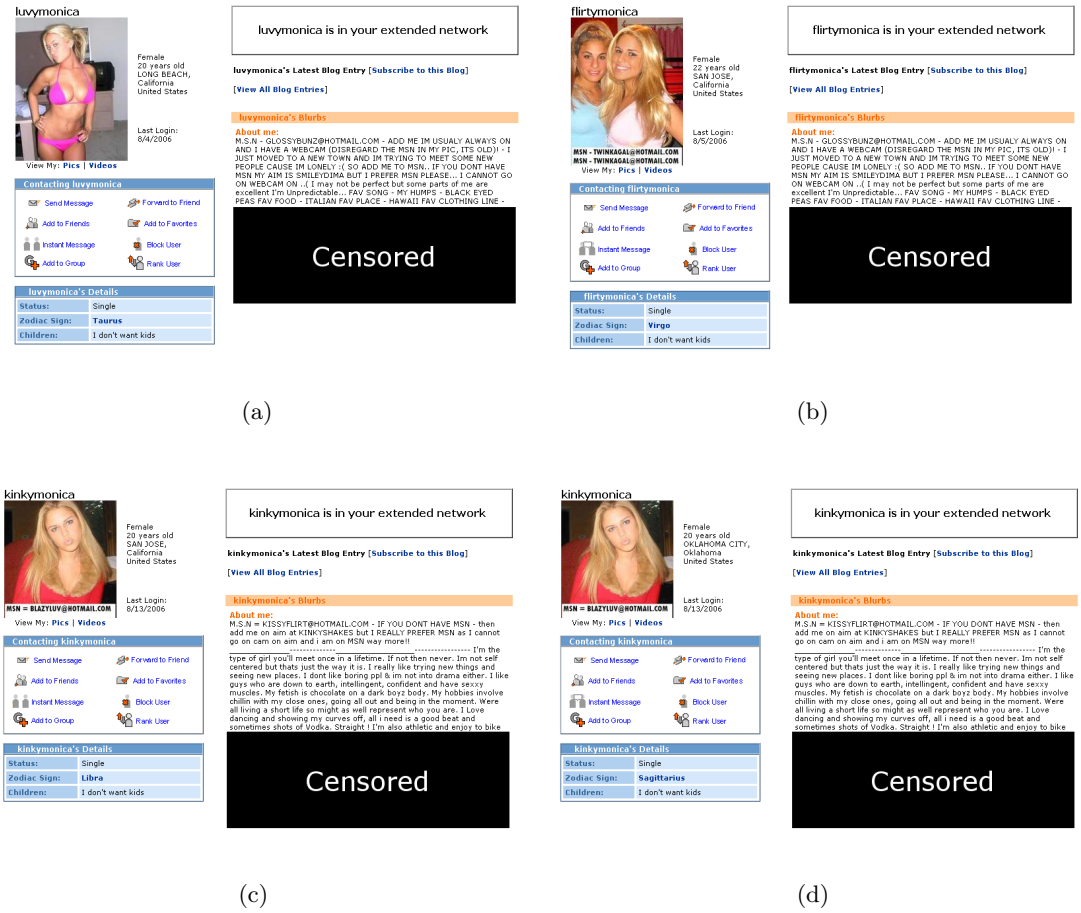


Figure 56: Pornographic rogue advertising profiles.

messenger screen names to induce users to contact IM bots. Finally, all of the profiles claim to be located in different cities. Despite their uncanny similarities, the profiles in Figures 56(a) and 56(b) are supposedly located in different parts of California, and the profiles in Figures 56(c) and 56(d) are in completely different parts of the country. These location differences exist because each profile is meant to target males in a different geographic area. This also explains why the profiles are allowed to be so similar. As long as the profiles in a given location are unique, users are less likely to become suspicious that the profiles are fraudulent.

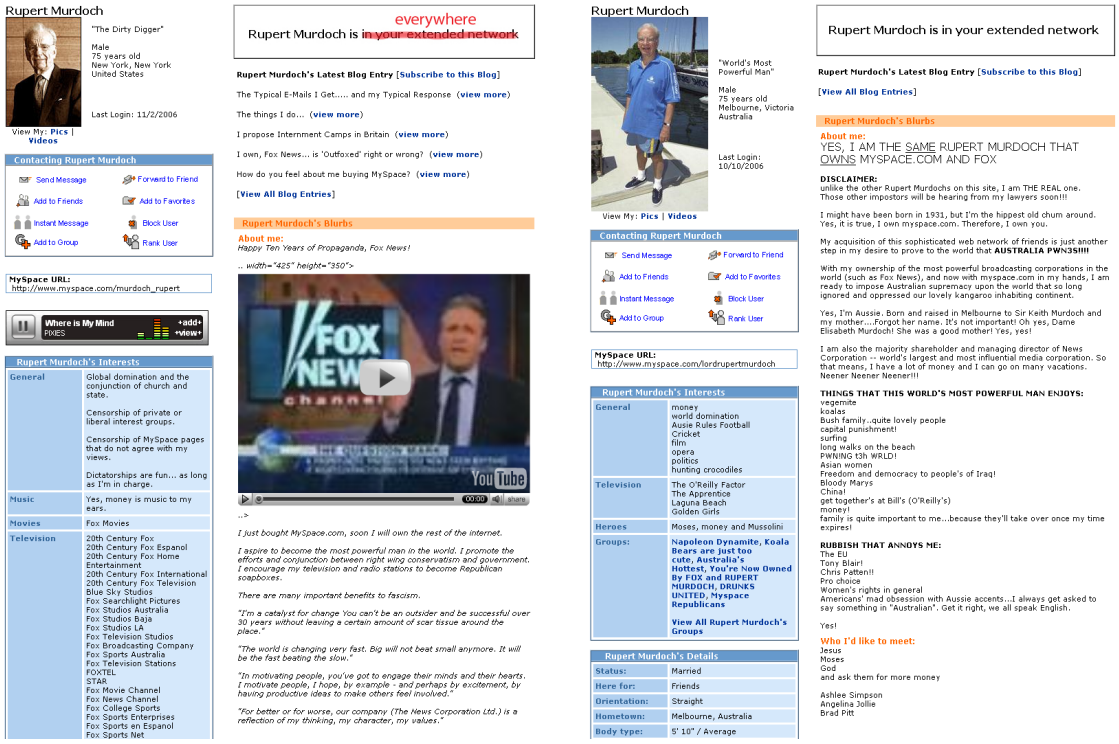
Deceptive rogue advertising profiles represent a serious security threat because they directly manipulate the behavior of users. Since the profiles appear to be maintained by

ordinary (albeit very attractive) people, other users mistakenly trust their content. Additionally, users often believe they have made a connection with the people pictured on these profiles because the users received friend requests from these profiles. Consequently, users access the URLs that are on the profiles (or in IM conversations) and become victims to the content found on the corresponding Web pages. In most cases, these pages contain pornographic or gambling-related material, but nothing prevents malicious individuals from embedding malware on the pages. Thus, to protect its users, social networking communities must develop techniques to identify and eliminate these profiles.

11.3.2 Impersonating Profiles

Due to the enormous popularity of social networking communities, they have emerged as forums for individuals to voice their opinions about various topics as well as other people. In MySpace, some users are more tactful and express their views in blog postings on their profiles or in comments on their friends' profiles. However, other users have taken their crusade to an entirely new level, creating new MySpace profiles that directly target specific ideals, companies, and even people. We refer to these profiles as *impersonating profiles* because they impersonate their targets to convey their message.

MySpace contains profiles that impersonate actors (e.g., Tom Hanks), athletes (e.g., Michael Jordan), technologists (e.g., Bill Gates), and politicians (e.g., George W. Bush). Even ancient philosophers, such as Socrates and Aristotle, have profiles dedicated to them. In fact, most of these individuals are impersonated by multiple distinct profiles. In some cases, these profiles are meant to be an homage to the individuals in question. However, more often than not, the impersonating profiles are meant to be slanderous to the targets of the impersonation. For example, after News Corporation purchased MySpace in July 2005, impersonating profiles for Rupert Murdoch (News Corporation's CEO) began to appear, making claims such as, "I just bought MySpace.com, soon I will own the rest of the internet" and "Dictatorships are fun... as long as I'm in charge." Two examples of these profiles are shown in Figures 57(a) and 57(b). To make matters worse, impersonators are able to use MySpace's communication facilities to contact other users under the guise of these profiles.



(a) (b)

Figure 57: Example impersonating profiles for Rupert Murdoch.

Thus, these counterfeit Rupert Murdoch profiles can express any number of scandalous opinions, and the negative backlash will be directed at the real Rupert Murdoch.

From the impersonators' standpoint, these profiles appear to be amusing satire. However, from the victims' standpoint, these profiles represent a form of identity theft as well as a public relations nightmare. The general public is unable to verify the authenticity of these profiles; thus, any slanderous comments found on the profiles (or received under the guise of the profiles) will be incorrectly associated with the victims of the impersonation. Using the example above, nothing prevents users from believing that Rupert Murdoch actually created one of those profiles or that he actually shares the viewpoints it contains. Admittedly, these viewpoints are somewhat absurd, but it is easy to envision subtler comments that can severely damage an individual's reputation. Since the success of most public figures is completely contingent upon the strength of their fan bases, an impersonating profile represents a direct threat to its victim's credibility and livelihood.

Sadly, this growing epidemic is not limited to high profile individuals. Private citizens such as teachers, principals, and police officers have also been victimized by these impersonating profiles. In fact, over the past couple of years, the news has been filled with stories about these types of attacks. For example, in December 2005, a 16-year-old boy posted an impersonating profile of a local police officer, which contained various derogatory statements about the police officer's appearance, intelligence, and sexual orientation [136]. Then, in April 2006, an eighth grader created impersonating profiles of his English teacher that contained racist remarks and falsely represented the teacher as a pornographer and child molester [131]. In September 2006, a high school assistant principal sued two students for creating an impersonating profile that falsely identified her as a lesbian [11]. These examples clearly illustrate the magnitude of this problem, and unfortunately, they represent a small sample of a growing list of incidents.

In addition to the risks already mentioned, impersonating profiles can also be used to amplify the severity of the other attacks we have already discussed. Specifically, a spammer could create an impersonating profile for a MySpace user and use the profile to spam that user's friends. Malware creators and phishers could also use this approach to spread their malicious content to the user's friends. Since these friends would be under the impression that they were communicating with the authentic profile, they would be much more likely to trust the communication, and as a result, the attacks would be far more successful.

11.4 Summary

In only a few years, social networking communities have made staggering strides in popularity (MySpace welcomed more than 65 million unique visitors in February 2008 [59]) and importance (YouTube was acquired by Google for $1.65 billion in October 2006 [67]). Aside from their social and economic impact, one of the most important questions is whether social networking environments are safe for their users. In this chapter, we have analyzed this safety question and described several security and privacy threats that have translated into real attacks in MySpace. Some of these attacks have been adapted from other Internet environments, including variants of malware propagation, spam, and phishing. Other

attacks are new and unique to social networking environments, including the creation of rogue advertising profiles and impersonating profiles. From our real world observations, it is clear that additional efforts should be made to protect social networking users.

CHAPTER XII

USING SOCIAL HONEYPOTS TO IDENTIFY SPAMMERS

Over the past few years, social networking communities have experienced unprecedented growth, both in terms of size and popularity. In fact, of the top-20 most visited World Wide Web destinations, six are now social networks, which is five more than the list from only three years ago [3]. This flood of activity is remaking the Web into a "social Web" where users and their communities are the centers for online growth, commerce, and information sharing. Unfortunately, as we found in the previous chapter, the rapid growth of these communities has made them prime targets for attacks by malicious individuals. Most notably, these communities are being bombarded by *social spam* [79].

Some of the spam in social networking communities is quite familiar. For example, message spam within a community is similar in form and function to email spam on the wider Internet, and comment spam on social networking profiles manifests itself in a similar fashion to blog spam. Defenses against these familiar forms of spam can be easily adapted to target their social networking analogs [79]. However, other forms of social spam are new and have risen out of the very fabric of these communities. One of the most important examples of this new generation of spam is deceptive spam profiles, which attempt to manipulate user behavior. These deceptive profiles are inserted into the social network by spammers in an effort to prey on innocent community users and to pollute these communities. Although fake profiles (or fakesters) have been a "fun" part of online social networks from their earliest days [148], growing evidence suggests that spammers are deploying deceptive profiles in increasing numbers and with more intent to do harm. For example, deceptive profiles can be used to drive legitimate users to Web spam pages, to distribute malware, and to disrupt the quality of community-based knowledge by spreading disinformation.

Understanding different types of social spam and deception is the first step towards countering these vulnerabilities. Hence, in this chapter, we propose a novel technique for

harvesting deceptive spam profiles from social networking communities using *social honeypots*. Then, we provide a characterization of the spam profiles that we collected with our social honeypots. To the best of our knowledge, this is the first use of honeypots in the social networking environment as well as the first characterization of deceptive spam profiles.

Our social honeypots draw inspiration from security researchers who have used honeypots to observe and analyze malicious activity. Specifically, honeypots have already been used to characterize malicious hacker activity [142], to generate intrusion detection signatures [99], and to observe email address harvesters [120]. In our current research, we create honeypot profiles within a community to attract spammer activity so that we can identify and analyze the characteristics of social spam profiles. Concretely, we constructed 51 honeypot profiles and associated them with distinct geographic locations in MySpace, the largest and most active social networking community. After creating our social honeypots, we deployed them and collected all of the traffic they received (via friend requests). Based on a four month evaluation period from October 1, 2007 to February 1, 2008, we have conducted a sweeping characterization of the harvested spam profiles from our social honeypots. A few of the most interesting findings from this analysis are:

- The spamming behaviors of spam profiles follow distinct temporal patterns.
- The most popular spamming targets are Midwestern states, and the most popular location for spam profiles is California.
- The geographic locations of spam profiles almost never overlap with the locations of their targets.
- 57.2% of the spam profiles obtain their "About me" content from another profile.
- Many of the spam profiles exhibit distinct demographic characteristics (e.g., age, relationship status, etc.).
- Spam profiles use thousands of URLs and various redirection techniques to funnel users to a hand full of destination Web pages.

The rest of the chapter is organized as follows. Section 12.1 summarizes related work. Section 12.2 provides background information about social networking communities and describes the social spam that is currently plaguing these communities. In Section 12.3, we present our methodology for creating social honeypots and collecting deceptive spam profiles. In Section 12.4, we report the results of an analysis we performed on the spam profiles that we collected in these honeypots. Section 12.5 summarizes our results.

12.1 Related Work

Due to the explosive growth and popularity of social networking communities, a great deal of research has been done to study various aspects of these communities. Specifically, these studies have focused on usage patterns [29, 63], information revelation patterns [29, 81], and social implications [44, 49] of the most popular communities. Work has also been done to characterize the growth of these communities [100] and to predict new friendships [102] and group formations [12].

Recently, researchers have also begun investigating the darker side of these communities. For example, numerous studies have explored the privacy threats associated with public information revelation in the communities [1, 12, 21, 72]. Aside from privacy risks, researchers have also identified attacks that are directed at these communities (e.g., social spam) [79]. In the previous chapter, we showed that social networking communities are susceptible to two broad classes of attacks: traditional attacks that have been adapted to these communities (e.g., malware propagation) and new attacks that have emerged from within the communities (e.g., deceptive spam profiles).

Unfortunately, very little work has been done to address the emerging security threats in social networking communities. Heymann et al. [79] presented a framework for addressing these threats, and Zinman and Donath [157] attempted to use machine learning techniques to classify profiles. However, the research community desperately needs real-world examples and characterizations of malicious activity to inspire new solutions. Thus, to help address this problem, we present a novel technique for collecting deceptive spam profiles in social networking communities that relies on social honeypot profiles. Additionally, we provide the

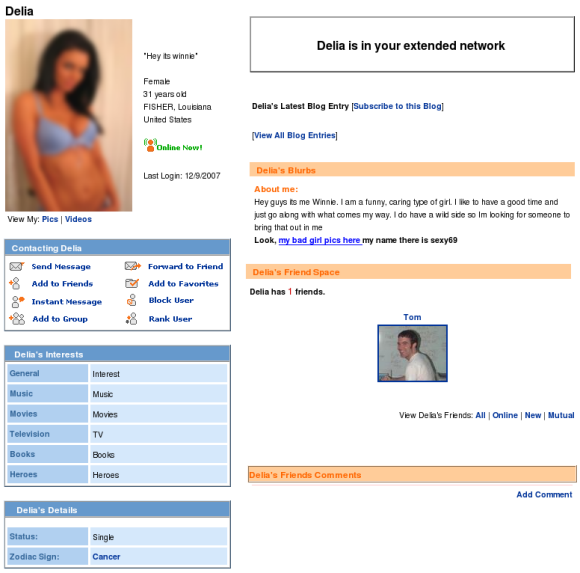


Figure 58: An example of a deceptive spam profile.

first characterization of deceptive spam profiles in an effort to stimulate research progress.

12.2 Social Spam

Social networking communities, such as MySpace, provide an online platform for people to manage existing relationships, form new ones, and participate in a variety of social interactions. To facilitate these interactions, a user's online presence in the community is represented by a *profile*, which is a user-controlled Web page that contains a picture of the user and various pieces of personal information. Additionally, a user's profile also contains a list of links to the profiles of that user's friends. Each of these friend links is bidirectional and established only after the user has received and accepted a *friend request* from another user.

Aside from friend requests, MySpace also provides a number of other communication facilities that enable users to communicate with each other within the community. These facilities include (but are not limited to) messaging, commenting, and blogging systems. Unfortunately, spammers have already begun exploiting these systems by propagating spam (e.g., message spam, comment spam, etc.) through them. Even more troubling is the fact that spammers are now polluting the communities with deceptive spam profiles that aim

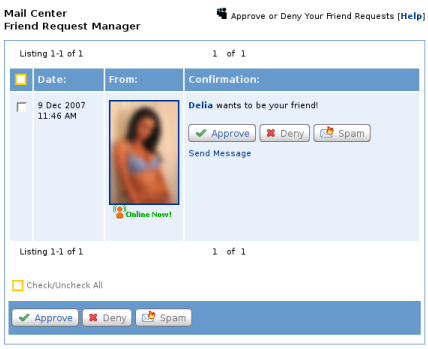


Figure 59: An example of a spam friend request.

to manipulate legitimate users.

An example of a MySpace spam profile[1] is shown in Figure 58. As the figure shows, spam profiles contain a wealth of information and various deceptive properties. Most notably, these profiles typically use a provocative image of a woman to entice users to view them. Then, once the profiles have attracted visitors, they direct those visitors to perform an action of some sort (e.g., visiting a Web page outside of the community) by using a seductive story in their "About me" sections. For example, the profile in Figure 58 provides a link to another Web page and promises that "bad girl pics" will be found there.

After spammers have constructed their deceptive profiles, they must attract visitors. To generate this traffic for their profiles, spammers typically employ two strategies. First, spammers keep their profiles logged in to MySpace for long periods of time. This strategy generates attention because many of the MySpace searching mechanisms give preferential treatment to profiles that are currently logged in to the system. Consequently, when users are browsing through profiles, the spam profiles will be prominently displayed. The second strategy is much more aggressive and involves sending friend requests to MySpace users. Figure 59 shows an example friend request that corresponds to the profile shown in Figure 58. Unlike the first strategy, which passively relies on users to visit the spam profiles, this strategy actively contacts users and deceives them into believing the profile's creator wants to befriend them.

[1] All of the provocative images in the chapter have been blurred so as not to offend anyone.

Figure 60: An example of a social honeypot.

12.3 Social Honeypots

For years, researchers have been deploying honeypots to capture examples of nefarious activities [99, 120, 142]. In this chapter, we utilize honeypots to collect deceptive spam profiles in social networking communities. Specifically, we created 51 MySpace profiles to serve as our social honeypots. To observe any geographic artifacts of spamming behavior, each of these profiles was given a specific geographic location (i.e., one honeypot was assigned to each of the U.S. states and Washington, D.C.). With the exception of the D.C. honeypot, each profile's city was chosen based on the most populated city in a given state. For example, Atlanta has the largest population in Georgia, and as a result, it was the city used for the Georgia honeypot. We used this strategy because we assumed that spammers would target larger cities due to their larger populations of potential victims.

All 51 of our honeypot profiles are identical except for their geographic information (see Figure 60 for an example). Each profile has the same name, gender, and birthday. Additionally, all of the demographic information was chosen to make the profiles appear attractive to spammers. Specifically, all of the profiles share the same relationship status (single), body type (athletic), and ethnicity (White / Caucasian). These demographic characteristics were also among the most popular in our previous large-scale characterization of MySpace profiles [29].

To collect timely information and increase the likelihood of being targeted by spammers, we created custom MySpace bots to ensure that all of our profiles are logged in to MySpace

24 hours a day, 7 days a week [2]. In addition to keeping our honeypot profiles logged in to the community, our bots also monitor any spamming activity that is directed at the profiles. Specifically, the bots are constantly checking the profiles for newly received friend requests. To avoid burdening MySpace with excessive traffic (and to avoid being labeled as a spam bot), each of our bots follows a polling policy that employs random sleep timers and an exponential backoff algorithm, which fluctuates sleep times based on the current amount of spamming activity (with a minimum and maximum sleep time of five minutes and one hour, respectively). Therefore, when a honeypot profile is receiving spam, its corresponding bot polls MySpace more aggressively than when the profile is not receiving spam.

After one of our honeypot profiles receives a new friend request, the bot responsible for that profile performs various tasks. First, the bot downloads the spam profile that sent the friend request[3], storing a copy of the profile along with a honeypot-specific identifier and a timestamp that corresponds to the time when the friend request was sent to the honeypot profile. Then, after storing a local copy of the profile, the bot rejects the friend request. We decided to reject the friend requests for two reasons. First, we wanted to identify spam profiles that are repeat offenders (i.e., they continuously send friend requests until they are accepted). Second, we did not want our honeypot profiles to be mistaken as spam profiles. If we blindly accept all of the spam friend requests, our honeypot profiles will appear to be helping the spam profiles in a manner similar to a Web spam page that participates in a link exchange or link farm [73]. Thus, to avoid suspicion by MySpace, our honeypot profiles conservatively reject the friend requests that they receive.

Many of the spam profiles contain links in their "About me" sections that direct users to Web pages outside of the social networking community. We wanted to study the characteristics of these Web pages; hence, in addition to storing the spam profiles, our bots also crawl the pages that are being advertised by these profiles. Specifically, after one of our bots stores a local copy of a profile, the bot parses the profile's "About me" section

[2]We experienced a few short outages (on the order of hours) due to MySpace system updates, which forced us to slightly modify our bots.

[3]Band profiles also send unsolicited friend requests; however, our bots simply reject these requests without processing them.

and extracts its URLs. Then, the bot crawls the Web pages corresponding to those URLs, storing them along with their associated spam profile.

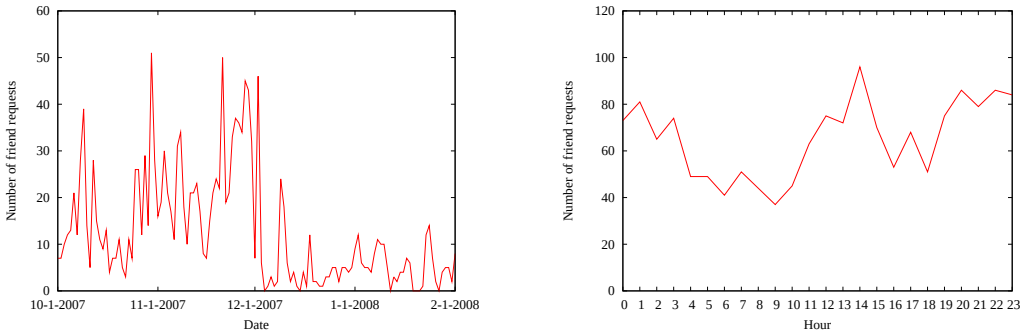
Not surprisingly, almost all of the URLs that are advertised by spam profiles are entrances to sophisticated redirection chains. To identify the final destinations in these chains, our bots follow every redirect. First, the bots attempt to access each of the URLs being advertised by a spam profile. If a bot encounters HTTP redirects (i.e., 3xx HTTP status codes), the bot follows them until it accesses a URL that does not return a redirect. Then, the corresponding Web page is stored and parsed for HTML/javascript redirection techniques using the redirection detection algorithm we presented in Chapter 9. If our algorithm extracts redirection URLs, our bots attempt to access them. Once again, if the bots encounter HTTP redirects, they follow the redirects until they find URLs that do not return redirects. Finally, the corresponding Web pages are stored. After completing this process, we are left with a collection of final destination pages and the intermediary pages that were crawled along the way.

12.4 Social Honeypot Data Analysis

In this section, we investigate various characteristics of the 1,570 friend requests (and corresponding spam profiles) that we collected in our social honeypots during a four month evaluation period from October 1, 2007 to February 1, 2008. First, we characterize the temporal distribution of the spam friend requests that our honeypots received. Then, we analyze the geographic properties of social spam. Next, we investigate duplication in spam profiles and identify five popular groups of spam profiles. After our duplication analysis, we identify interesting demographic characteristics of spam profiles. Finally, we analyze the Web pages that are advertised by spam profiles.

12.4.1 Temporal Distribution of Spam Friend Requests

Since all of our honeypot profiles were constantly logged into MySpace during our four month evaluation period, we were able to observe various temporal patterns for spamming activity. In Figure 61(a), we present the number of friend requests that our honeypots received on each day of this four month period. This figure is interesting for a few reasons.



(a) Monthly distribution

(b) Hourly distribution

Figure 61: Temporal distributions of the spam friend requests received by our social honeypots.

First, we observe three peaks in spamming activity that occur around holidays. Specifically, our honeypots received the most friend requests the day before, the day of, and the day after Columbus Day (79), Halloween (95), and Thanksgiving (90). One possible explanation for these spikes is that legitimate users might spend more time online during these holiday periods, giving spammers a larger audience for their deceptive profiles.

Another intriguing observation from Figure 61(a) is that our honeypots began receiving significantly fewer friend requests after December 2. In fact, of the 1,570 friend requests that our honeypots received, only 299 (19.0%) of them were received after this date. We are still investigating the reasons behind this reduced activity, but one hypothesis is that spammers realized the underlying purpose of our honeypot profiles. Since all of our honeypots reject friend requests after the corresponding spam profiles have been stored, spammers should eventually recognize that each of the honeypots represents a wasted friend request. As we explained in Section 12.3, we decided to reject spam friend requests because we wanted to avoid having our honeypots labeled as spam by MySpace. As part of our ongoing research, we are revisiting this decision to investigate whether it affects the spamming activity we observe.

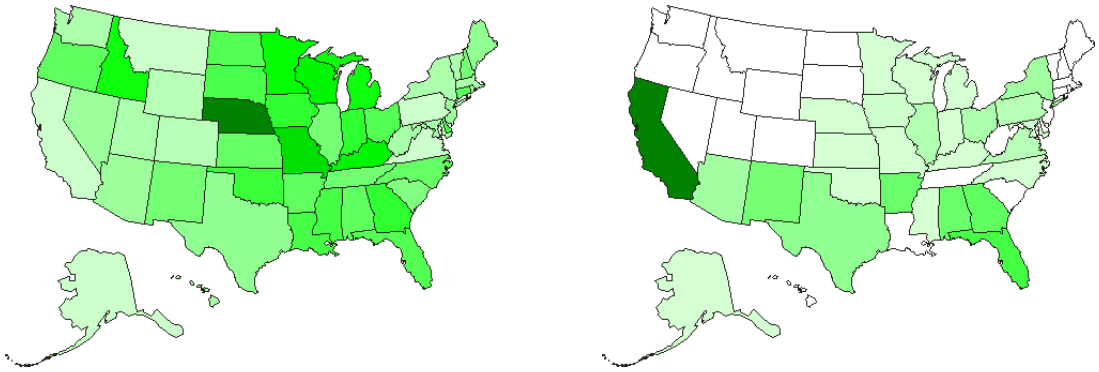
To analyze finer-grained temporal patterns, Figure 61(b) shows the hourly distribution

of the friend requests that our honeypots received[4]. As the figure shows, for every hour of the day, our honeypots received at least 35 friend requests from spammers. Additionally, distinct hourly patterns emerge from the figure. Most notably, spamming activity is at its peak around 2pm and from 10pm to 1am, and it is at its lowest levels between 4am and 9am. These patterns are particularly interesting because they mirror previous results about the communication patterns of legitimate users in social networking communities [63]. The similarities between legitimate and spam activity patterns are somewhat intuitive for at least two reasons. First, spammers want to be active when their targets are active because they want to increase the chances of successfully deceiving those users. Second, by blending their traffic in with legitimate traffic, spammers reduce the risk of being identified by the operators of these communities.

12.4.2 Geographic Distribution of Spam Friend Requests

Since each of our honeypot profiles claims to be in a unique geographic location (i.e., one honeypot is in each of the fifty U.S. states and Washington, D.C.), we are able to analyze the geographic properties of spamming behavior. Figure 62(a) shows a color-coded map of the United States, which represents the relative popularity of our geographically dispersed honeypots. States with darker shades of green represent honeypots that received more friend requests than the honeypots in states with lighter shades of green. As the figure shows, a large fraction of the spamming activity was directed at the Midwestern states. In fact, the five most popular targets were the honeypots in Omaha, Nebraska (80 friend requests), Kansas City, Missouri (58 friend requests), Milwaukee, Wisconsin (56 friend requests), Louisville, Kentucky (56 friend requests), and Minneapolis, Minnesota (53 friend requests). In our previous research [29], we found that MySpace users from Midwestern states began using MySpace considerably later than users from Western states because MySpace was founded in California. As a result, one explanation for why Midwestern users are frequently targeted by spammers is that those users might be less MySpace-savvy and thus a more attractive target for deception by spammers. This hypothesis is also supported by the fact

[4]All of the times are normalized based on the time zone that corresponds to the honeypot profile's location.



(a) Target distribution

(b) Originator distribution

Figure 62: Geographic distributions of spam profiles and their targets.

that our Los Angeles, California honeypot received fewer friend requests (10) than any of the other honeypots.

Figure 62(b) shows another color-coded map of the United States. However, unlike Figure 62(a), this figure shows the relative popularity of various states as locations for spam profiles. States with darker shades of green have more spam profiles affiliated with them than states with lighter shades of green. As the figure shows, the most popular locations for the spam profiles were California and Southeastern states. Specifically, the five most popular states for spam profiles were California (186 spam profiles), Florida (92 spam profiles), Georgia (78 spam profiles), Arkansas (74 spam profiles), and Alabama (73 spam profiles).

After investigating the most popular originating and target locations for spam profiles, an obvious question is how much overlap (if any) exists for those locations. Concretely, we wanted to know how often the declared location of a spam profile matches the declared location of a targeted profile. Our original hypothesis was that we would identify a significant number of matches because we believed victims might be hesitant to accept a friend request from someone outside of their city or state. However, we were surprised to find that 1,534 (97.7%) of the friend requests were from spam profiles that reported a location that did not match the city or state associated with the honeypot profile that received them.

One explanation for this disconnect between the locations of spam profiles and their

victims is that a clear tension exists between increasing the deceptive properties of a spam profile and making the profile broadly applicable to a large number of potential victims. Obviously, spammers would prefer to create personalized spam profiles for every potential victim because that would greatly increase the likelihood of a successful deception. However, it is costly to create personalized profiles for every potential victim, and as a result, spammers focus on casting as wide a net as possible.

12.4.3 Spam Profile Duplication

While investigating the geographic distribution of the friend requests that our honeypot profiles received, we noticed that many of our honeypots received a friend request from the same spam profile. In fact, 65 spam profiles sent a friend request to more than one of our honeypots, generating a total of 148 friend requests. 40 (78.4%) of our honeypots received at least one of these duplicate friend requests, and the honeypots that received the most friend requests (i.e., the Omaha, Nebraska honeypot and the Kansas City, Missouri honeypot) also received the most duplicates (11 duplicates and 8 duplicates, respectively). Surprisingly, none of our honeypots received more than one friend request from a given spam profile (i.e., none of the spam profiles were repeat offenders). Thus, after one of our honeypots rejected a spam profile's friend request, that profile was intelligent enough not to send the honeypot another friend request.

After we identified the existence of duplicate friend requests, we wanted to determine how much lag time (if any) exists between the first arrival of a friend request and the arrivals of its duplicates. To quantify the delays between duplicate friend requests, we created 65 time series – one for each set of the duplicate friend requests. Then, for each time series, we computed the size of the time window that includes the first and last point. Based on this analysis, we found that 63 (96.9%) of the time windows close in less than 4 minutes, and 53 (81.5%) of the time windows close in less than a minute. Therefore, when these spam profiles sent friend requests, they sent a large number of them in a short period of time (i.e., they were not particularly stealthy).

Once we determined the number of unique profiles in our collection (1,487), we wanted

to know how many of those profiles possess content that is a duplicate (or a near-duplicate) of another profile's content. In Chapter 9, we found that only one-third of Web spam pages are unique, and we wanted to determine if the same level of duplication exists among spam profiles. To quantify the amount of content duplication in our collection of 1,487 unique spam profiles, we used the shingling algorithm from Chapter 9 on all of their HTML content to construct equivalence classes of duplicate and near-duplicate profiles.

First, we preprocessed each profile by replacing its HTML tags with white space and tokenizing it into a collection of words (where a word is defined as an uninterrupted series of alphanumeric characters). Then, for every profile, we created a fingerprint for each of its n words using a Rabin fingerprinting function [124] (with a degree 64 primitive polynomial p_A). Once we had the n word fingerprints, we combined them into 5-word phrases. The collection of word fingerprints was treated like a circle (i.e., the first fingerprint follows the last fingerprint) so that every fingerprint started a phrase, and as a result, we obtained n 5-word phrases. Next, we generated n phrase fingerprints for the n 5-word phrases using a Rabin fingerprinting function (with a degree 64 primitive polynomial p_B). After we obtained the n phrase fingerprints, we applied 84 unique Rabin fingerprinting functions (with degree 64 primitive polynomials p_1, ..., p_{84}) to each of the n phrase fingerprints. For every one of the 84 functions, we stored the smallest of the n fingerprints, and once this process was complete, each spam profile was reduced to 84 fingerprints, which are referred to as that profile's *shingles*. Once all of the profiles were converted to a collection of 84 shingles, we clustered the profiles into equivalence classes (i.e., clusters of duplicate or near-duplicate profiles). Two profiles were considered duplicates if all of their shingles matched, and they were near-duplicates if their shingles agreed in two out of the six possible non-overlapping collections of 14 shingles. For a more detailed description of this shingling algorithm, please consult [24, 55].

After this clustering was complete, we were left with 1,261 unique clusters of duplicate and near-duplicate profiles. Thus, only 226 (15.2%) of the profiles have the same (or nearly the same) HTML content as one of the remaining 1,261 profiles. This level of duplication is significantly less than what we observed with Web spam pages; however, we do not

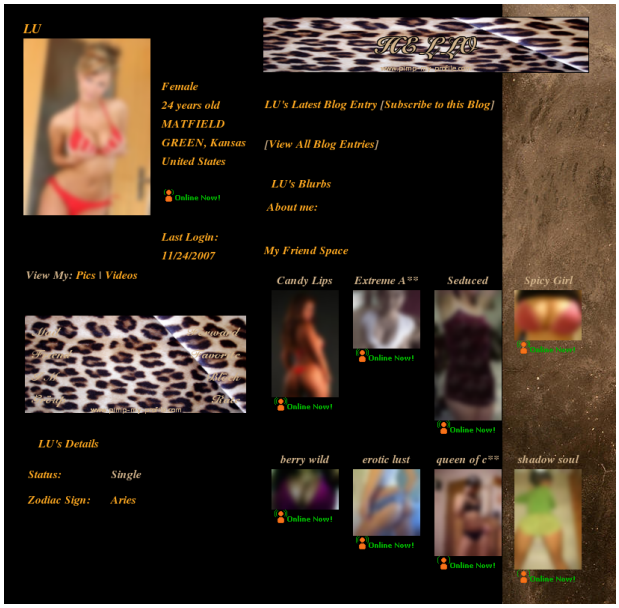


Figure 63: An example of a Click Trap.

believe this is an accurate measure of spam profile duplication. Since most of a spam profile's deceptive text is found in the "About me" section, a more reasonable metric for profile duplication is actually "About me" duplication. Hence, in addition to running our shingling algorithm over all of the HTML content in a profile, we also extracted the "About me" content and built equivalence classes using that data. This "About me" clustering generated 637 unique clusters, which means 850 (57.2%) of the profiles have the same (or nearly the same) "About me" content as one of the remaining 637 profiles.

Based on the results of our content duplication analysis, we can conclude that duplication among spam profiles is on par with duplication among Web spam pages. 15.2% of the spam profiles obtain all of their HTML content from another profile, and 57.2% of the spam profiles obtain their "About me" content from another profile. This observation is quite encouraging because it implies that the problem of identifying all spam profiles can actually be reduced to the problem of identifying a much smaller set of unique profiles.

12.4.4 Spam Profile Examples

After we completed our content duplication analysis, we manually investigated the profiles in our various clusterings. Based on this investigation, we discovered that most of our spam

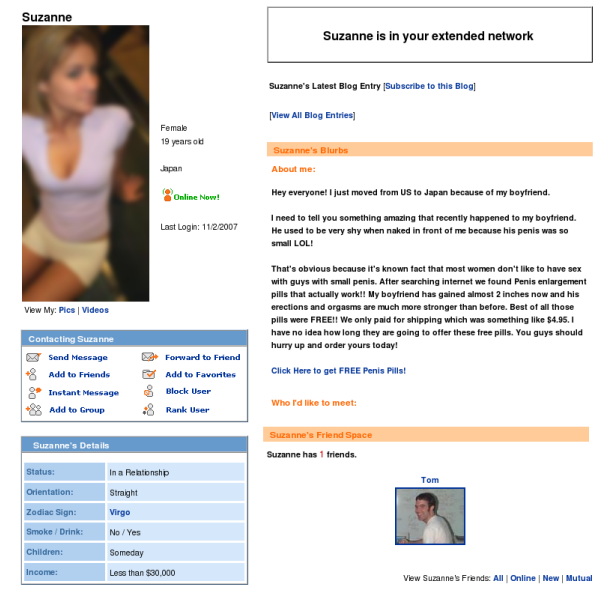


Figure 64: An example of a Japanese Pill Pusher.

profiles fall into one of five categories:

- **Click Traps**: Each profile contains a background image that is also a link to another Web page. If users click anywhere on the profile, they are immediately directed to the link's corresponding Web site. One of the most popular (and most deceptive) examples displays a fake list of friends, which is actually a collection of provocative images that direct users to a nefarious Web page (see Figure 63 for an example).

- **Friend Infiltrators**: These profiles do not have any overtly deceptive elements (aside from their images – and even those are innocuous in some cases). The purpose of the profiles is to befriend as many users as possible so that they can infiltrate the users' circles of friends and bypass any communication restrictions imposed on non-friends. Once a user accepts a friend request from one of these profiles, the profile begins spamming that user through every available communication system (e.g., message spam, comment spam, etc.).

- **Pornographic Storytellers**: Each of these profiles has an "About me" section that consists of randomized pornographic stories, which are bookended by links that lead to pornographic Web pages. The anchor text used in these profiles is extremely similar,

even though the rest of the "About me" text is almost completely randomized.

- **Japanese Pill Pushers**: These profiles contain a sales pitch for male enhancement pills in their "About me" sections. According to the pitch, the attractive woman pictured in the profile has a boyfriend that purchased these pills at an incredible discount, and if you act now, you can do the same. An example is shown in Figure 64.

- **Winnies**: All of these profiles have the same headline: "Hey its winnie." However, despite this headline, none of the profiles are actually named "Winnie." In addition to a shared headline, each of the profiles also includes a link to a Web page where users can see the pictured female's pornographic pictures. An example of one of these profiles was shown in Section 12.2 (Figure 58).

12.4.5 Spam Profile Demographics

In our content duplication analysis, we analyzed the HTML and "About me" sections of spam profiles in a general sense. To observe more specific features of these profiles, we investigated demographic characteristics of the 1,487 spam profiles that we captured in our honeypots. These characteristics include traditional demographics (e.g., age, gender, etc.) as well as profile-specific features (e.g., number of friends, headlines, etc.).

Our first observation from this demographic analysis is that many of the spam profiles share various demographic characteristics. Specifically, all of the profiles are female and between the ages of 17 and 34 (85.9% of the profiles state an age between 21 and 27). Additionally, 1,476 (99.3%) of the profiles report that they are single. None of these characteristics are particularly surprising because they all reinforce the deceptive nature of these profiles. Specifically, these demographic features make each profile appear as though it was created by a young, "available" woman.

Our second observation is that many spam profiles include additional personal information to enhance their deceptive properties. The profiles that are most adept at leveraging personal information to their advantage are the Japanese Pill Pushers. These profiles are the only ones that claim to be in a relationship, but this relationship status is warranted because the profiles mention a boyfriend in their "About me" sections. Additionally, these

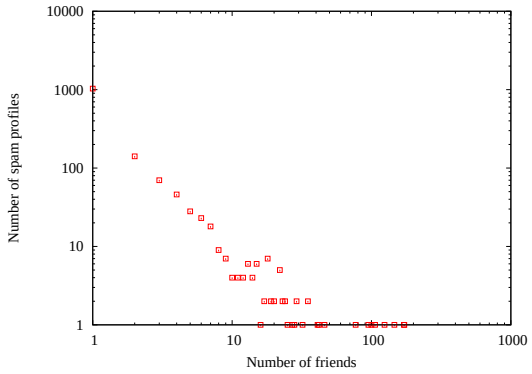


Figure 65: Distribution of the number friends associated with spam profiles.

profiles list "less than $30,000" as their annual income. This is the lowest allowable option on MySpace, and as a result, this annual income value makes it seem like the profile's creator is not particularly wealthy, which reinforces the affordability of the male enhancement pills that these profiles are advertising.

Our final observation is that many of the spam profiles successfully befriended legitimate users. Figure 65 shows the distribution of friends associated with each of the spam profiles. It is important to note that this distribution is skewed towards the low end of the spectrum because our bots visited and stored the spam profiles very quickly (and potentially before any other users had a chance to accept the spam friend requests). However, despite this fact, 455 (30.6%) of the profiles had more than one friend when our bots collected them. Thus, almost a third of the spam profiles were already attracting victims when our bots visited them.

12.4.6 Advertised Web Pages

Since the purpose of spam profiles is to deceive users into performing an action (e.g., visiting a Web page), most of the profiles contain links to Web pages outside of the community. Specifically, 1,245 (83.7%) of the profiles contain at least one link in their "About me" sections. The remaining 242 profiles are all examples of Friend Infiltrators, and as a result, they postpone their promotional activities until after they have befriended users.

From the 1,245 profiles that contain links, our bots were able to extract and successfully

access 1,048 *profile URLs*. Of these 1,048 profile URLs, only 482 (46.0%) of them were unique, which means more than half of the URLs that appear in spam profiles are duplicates. When our bots attempted to crawl the profile URLs, 339 (32.3%) of them returned a total of 657 HTTP redirects. After following these HTTP redirect chains, our original 482 unique profile URLs funneled our bots to only 148 unique destination URLs. Therefore, of the 1,048 Web pages that our bots ultimately obtained with the profile URLs, 900 (85.9%) of them have duplicate URLs.

To investigate this duplication even further, we performed a shingling analysis on the HTML content of these 1,048 Web pages. Based on this analysis, we discovered only 6 unique clusters of duplicate and near-duplicate Web pages. Thus, 1,042 (99.4%) of the Web pages contain content that was duplicated from the other 6 Web pages. Three of these clusters, which account for 93.3% of the pages, contain pages that act as intermediary redirection pages (i.e., the pages immediately redirect users using HTML/javascript redirection techniques). Two of the clusters, which account for 6.6% of the pages, contain pornographic Web pages, and the last cluster contains a single Web page, which executes a phishing attack against MySpace.

Since 93.3% of the pages employ redirection techniques, we parsed those pages for HTML/javascript redirects using our redirection detection algorithm from Chapter 9. Based on this redirection analysis, we identified redirects that use HTML meta refresh tags, javascript location variable assignments, and HTML iframe tags. In total, our algorithm identified 1,307 *redirection URLs*. However, of those 1,307 URLs, only 136 (10.4%) of them were unique; hence, over 90% of the redirection URLs are duplicates.

When our bots crawled these redirection URLs, 959 (73.4%) of them returned a total of 1,288 HTTP redirects. After following these HTTP redirect chains, our bots were eventually funneled to only 15 unique URLs. Thus, of the 1,307 Web pages that our bots crawled using the redirection URLs, only 15 (1.1%) of them have unique URLs. Even more striking is the fact that only 5 domain names are used in those 15 unique URLs, and of those 5 domain names, `fling.com` and `amateurmatch.com` Web pages account for 975 (74.6%) of the Web pages. An example of one of the `amateurmatch.com` Web pages is shown in Figure 66.

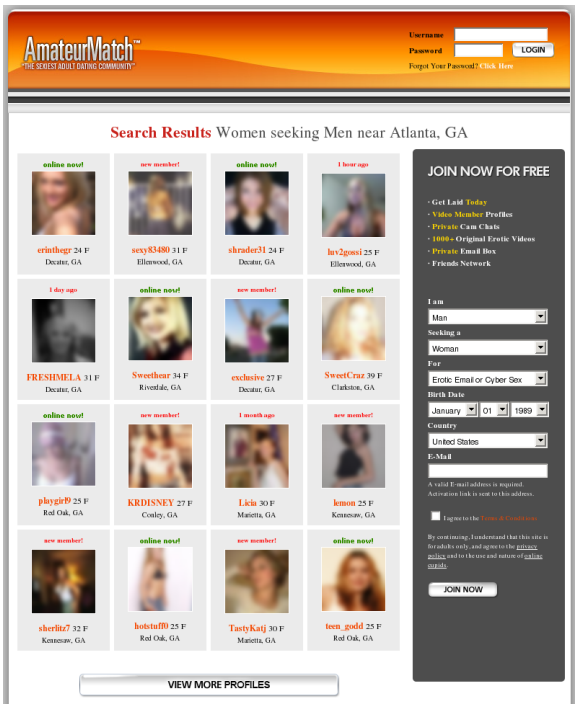


Figure 66: An example of a Web page that is advertised by a spam profile.

Based on the results of our Web page analysis, we can conclude that all of the URLs that are advertised in spam profiles point to an extremely small number of destination pages. Specifically, 1,048 profile URLs funneled our bots to only 6 destinations, and 1,307 redirection URLs funneled our bots to only 5 destinations. This observation is quite valuable because it significantly reduces the problem of identifying the Web pages that are advertised by spam profiles. Instead of dealing with 2,355 URLs, we must simply identify 11 destinations.

12.5 Summary

In this chapter, we presented a novel technique for automatically collecting deceptive spam profiles in social networking communities. Specifically, our approach deploys honeypot profiles and collects all of the spam profiles associated with the spam friend requests that they receive. We also provided the first characterization of deceptive spam profiles using the data that we collected in our 51 social honeypots over a four month evaluation period. This characterization covered various topics including temporal and geographic distributions of spamming activity, content duplication, an analysis of profile demographics, and an

evaluation of the Web pages that are advertised by spam profiles.

CHAPTER XIII

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis research, we focused on countering DoI attacks in three information-rich environments: email systems, the World Wide Web, and social networking communities. For each environment, we performed large-scale data collection and analysis operations to create massive corpora of low and high quality information. Then, we used our collections to identify characteristics that uniquely distinguish examples of low and high quality information. Finally, we used our characterizations to create techniques that automatically detect and remove low quality information from online information-rich environments.

13.1 Countering DoI Attacks in Email Systems

Our first contribution in this dissertation was a collection of techniques for automatically detecting and removing low quality information associated with DoI attacks in email systems. We began by performing a large-scale experimental evaluation of statistical spam classifier effectiveness. This evaluation showcased the importance of using large corpora for classifier evaluations, and it showed that classifiers are able to successfully distinguish between large corpora of low quality email messages. We then showed that spammers can significantly degrade the performance of these classifiers with camouflaged spam content. However, we were able to restore most of the accuracy for the classifiers by retraining them to identify camouflaged messages as spam.

Next, we identified a clear tension between spam producers and information consumers. Spam producers are constantly evolving their techniques to ensure their spam messages are delivered, and information consumers are constantly evolving their countermeasures to ensure they don't receive spam messages. To experimentally model this arms race, we investigated the evolution of various email spam construction techniques and found numerous techniques that were ineffective over time due to anti-spam countermeasures. We also identified examples of techniques that were coexisting with anti-spam countermeasures.

This latter group of techniques led us to revisit the problems associated with camouflaged spam content. Since spammers continually evolve their techniques, we believed they would also evolve their camouflaged messages to defeat statistical spam classifiers.

To test the effects of evolving camouflaged spam content, we evaluated the classifier retraining solution against progressively more advanced camouflaged messages. Based on this evaluation, we discovered that the retraining process is only temporarily effective against camouflaged messages. As information consumers evolve and retrain their classifiers, spammers construct new camouflaged messages, which represent a new generation of attacks that defeat the retrained classifiers. This process continues until both parties are firmly entrenched in a spam arms race. Fortunately, we proposed two solutions that allow information consumers to break free of this arms race. The first solution alters the statistical classifier training process by associating disproportionate weights to spam and legitimate features, and the second solution incorporates various non-textual email features (e.g., URLs) to significantly enhance the robustness of the spam classification process.

Moving forward, our research in this domain can extend along numerous dimensions. In the short term, we will focus on answering various open questions that were posed throughout this thesis research. For example, in Chapter 5, each one of the spamicity tests showing co-existence is a challenge to be explained in more detail since those filters were unable to "kill off" that particular spam construction technique. More concretely, the several conjectures and potential explanations for the interactions between a spamicity test and its associated spam construction technique should be verified quantitatively. Another interesting research question is the lack of extinction examples for collaborative filtering, despite the large number of extinctions. Is it possible that collaborative filtering approaches have some inherent limitations (e.g., time lag) that prevent them from causing any strain of spam to become extinct? For the long term, other attacks on statistical spam classifiers and defense mechanisms may engage in arms races similar to the camouflage attacks described in Chapters 4 and 6. A general theoretical framework analogous to adversarial classification, which allows us to escape endless arms race cycles, would be a significant research challenge.

13.2 Countering DoI Attacks in the World Wide Web

After successfully defending against DoI attacks in email systems, we presented a framework for collecting, analyzing, and classifying examples of DoI attacks in the World Wide Web. First, we leveraged an interesting link between email spam and Web spam to propose a novel technique for extracting large Web spam samples from the Web. Then, we used our automated Web spam collection method to create the Webb Spam Corpus – a first-of-its-kind, large-scale, and publicly available Web spam data set. The corpus consists of nearly 350,000 Web spam pages, making it more than two orders of magnitude larger than any other previously cited Web spam data set.

Then, we utilized the Web spam pages in the Webb Spam Corpus to perform the first large-scale characterization of Web spam using content and HTTP session analysis techniques. Our content analysis results are consistent with the hypothesis that Web spam pages are different from legitimate Web pages, showing far more duplication of physical content and URL redirections. Additionally, our content analysis offers a categorization of Web spam pages, which includes **Ad Farms**, **Parked Domains**, **Advertisements**, **Pornography**, and **Redirection**. Next, an analysis of session information collected during the crawling of the Webb Spam Corpus shows significant concentration of hosting IP addresses in two narrow ranges as well as significant overlaps among session header values.

Leveraging the results of our HTTP session analysis, we presented a lightweight, predictive approach to Web spam classification that relies exclusively on HTTP session information (i.e., hosting IP addresses and HTTP session headers). Concretely, we built an HTTP session classifier based on our predictive technique, and by incorporating this classifier into HTTP retrieval operations, we are able to detect Web spam pages before the actual content transfer. By applying our predictive technique to a corpus of almost 350,000 Web spam instances and almost 400,000 legitimate instances, we were able to successfully detect 88.2% of the Web spam pages with a false positive rate of only 0.4%. Additionally, our experiments show that our approach saves an average of 15.4 KB of bandwidth and storage resources for every successfully identified Web spam page, while only adding an average of 101μs to each HTTP retrieval operation.

Our predictive approach to Web spam classification is complementary to previous Web spam research that is focused on link-based and content-based analysis techniques. Thus, an interesting direction for future research involves combining our HTTP session classification process with these existing analysis techniques to create a multi-layered defense against Web spam. This multi-layered defense could potentially minimize the weaknesses of each individual technique, while offering additional robustness to the overall spam classification process.

13.3 Countering DoI Attacks in Social Environments

Our final contribution in this dissertation was a collection of techniques that detect and help prevent DoI attacks within social environments (particularly social networking communities). First, we showed that social networking communities are susceptible to numerous attacks, making it difficult for users to access high quality information in these environments. Specifically, we identified two attack classes: traditional attacks that have been adapted to these communities (e.g., malware propagation, spam, and phishing) and new attacks that have emerged through malicious social networking profiles (e.g., rogue advertising profiles and impersonating profiles). Concretely, we described examples of these attack types that are observable in MySpace, which is currently the most popular social networking community.

After we described the various security threats that exist in social environments, we focused our attention on social spam. Unfortunately, little is known about social spammers, their level of sophistication, or their strategies and tactics. Thus, we offered a novel technique for capturing examples of social spam, and we provided the first characterization of social spammers and their behaviors. Concretely, we made two contributions: (1) we introduced social honeypots for tracking and monitoring social spam, and (2) we reported the results of an analysis performed on spam data that was harvested by our social honeypots. Based on our analysis, we found that the behaviors of social spammers exhibit recognizable temporal and geographic patterns and that social spam content contains various distinguishing characteristics.

As part of our ongoing work, we will investigate additional techniques for automatically countering DoI attacks in this increasingly important domain. By utilizing the successful approaches we developed in other information-rich environments, we expect to observe high degrees of accuracy when detecting DoI attacks in social environments. Specifically, we will apply statistical classification techniques that rely on the features we discovered as part of our characterization work, and we will continue to research new methodologies that are resilient against constantly evolving adversaries.

REFERENCES

[1] Acquisti, A. and Gross, R., "Imagined communities: Awareness, information sharing, and privacy on the facebook," in *Proceedings of the 6th Workshop on Privacy Enhancing Technologies (PET '06)*, pp. 36 – 58, 2006.

[2] Ahamad, M. and others, "Guarding the Next Internet Frontier: Countering Denial of Information Attacks," in *Proceedings of the New Security Paradigms Workshop (NSPW '02)*, 2002.

[3] Alexa, "Alexa top 500 sites." `http://www.alexa.com/site/ds/top_sites?ts_mode=global`, 2008.

[4] Amitay, E. and others, "The Connectivity Sonar: Detecting Site Functionality by Structural Patterns," in *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HYPERTEXT '03)*, pp. 38–47, 2003.

[5] Anderson, D. S. and others, "Spamscatter: Characterizing Internet Scam Hosting Infrastructure," in *Proceedings of 16th Usenix Security Symposium (Security '07)*, 2007.

[6] Androutsopoulos, I. and others, "An Evaluation of Naive Bayesian Anti-Spam Filtering," in *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning*, pp. 9–17, 2000.

[7] Androutsopoulos, I. and others, "An Experimental Comparison of Naive Bayesian and Keyword-based Anti-spam Filtering with Encrypted Personal E-mail Messages," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 160–167, 2000.

[8] Androutsopoulos, I. and others, "Learning to Filter Spam E-mail: A Comparison of a Naive Bayesian and a Memory-based Approach," in *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 1–13, 2000.

[9] Androutsopoulos, I., Paliouras, G., and Michelakis, E., "Learning to Filter Unsolicited Commercial E-mail," Tech. Rep. 2004/2, National Center for Scientific Research "Demokritos", 2004.

[10] Apte, C., Damerau, F., and Weiss, S. M., "Automated Learning of Decision Rules for Text Categorization," *Information Systems*, vol. 12, no. 3, pp. 233–251, 1994.

[11] Associated Press, "Official sues students over myspace page." `http://www.sfgate.com/cgi-bin/article.cgi?file=/news/archive/2006/09/22%/national/a092749D95.DTL`, 2006.

[12] Backstrom, L., Dwork, C., and Kleinberg, J., "Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography," in *Proceedings of the 16th International Conference on the World Wide Web*, pp. 181 – 190, 2007.

[13] Banko, M. and Brill, E., "Mitigating the Paucity-of-Data Problem: Exploring the Effect of Training Corpus Size on Classifier Performance for Natural Language Processing," in *Proceedings of the 1st International Conference on Human Language Technology Research*, pp. 1–5, 2000.

[14] Banko, M. and Brill, E., "Scaling to Very Very Large Corpora for Natural Language Disambiguation," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL '01)*, pp. 26–33, 2001.

[15] Becchetti, L. and others, "Link-based characterization and detection of web spam," in *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '06)*, 2006.

[16] Bekkerman, R., McCallum, A., and Huang, G., "Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora," tech. rep., 2004.

[17] Benczur, A. A., Csalogany, K., Sarlos, T., and Uher, M., "Spamrank - fully automatic link spam detection," in *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, 2005.

[18] Berger, A. L., Della Pietra, S. A., and Della Pietra, V. J., "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[19] Berners-Lee, T., Masinter, L., and McCahill, M., "RFC 1738 - Uniform Resource Locators (URL)." `http://www.faqs.org/rfcs/rfc1738.html`, 1994.

[20] Boyd, C., "Teenagers used to push zango on myspace." `http://www.vitalsecurity.org/2006/07/teenagers-used-to-push-zango-on.ht%ml`, 2006.

[21] Boyd, D., "Social network sites: Public, private, or what?," 2007.

[22] Brightmail, "Spam Percentages and Spam Categories." `http://www.nospam-pl.net/pub/brightmail.com/spamstats_Dec2003.html`, 2003.

[23] Brightmail, "Spam Percentages and Spam Categories." `http://www.nospam-pl.net/pub/brightmail.com/spamstats_March2004.html`, 2004.

[24] Broder, A. and others, "Syntactic clustering of the web," in *Proceedings of the 6th International World Wide Web Conference (WWW '97)*, pp. 391–404, 1997.

[25] Burton, B., "SpamProbe Version 1.1x5." `http://sourceforge.net/projects/spamprobe/`, 2004.

[26] Carreras, X. and Marquez, L., "Boosting Trees for Anti-Spam Email Filtering," in *Proceedings of the 4th International Conference on Recent Advances in Natural Language Processing (RANLP '01)*, pp. 58–64, 2001.

[27] Castillo, C. and others, "A reference collection for web spam," *SIGIR Forum*, vol. 40, no. 2, pp. 11–24, 2006.

[28] Castillo, C. and others, "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, 2007.

[29] Caverlee, J. and Webb, S., "A large-scale study of myspace: Observations and implications for online social networks," in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM '08)*, 2008.

[30] Caverlee, J., Webb, S., and Liu, L., "Spam-resilient web rankings via influence throttling," in *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS '07)*, 2007.

[31] Chan, J., "SURBL - Spam URI Realtime Blocklists." `http://www.surbl.org/`, 2004.

[32] Chandrinos, K. V. and others, "Automatic web rating: Filtering obscene content on the web," in *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries (ECDL '00)*, pp. 403–406, 2000.

[33] Cho, J. and Roy, S., "Impact of web search engines on page popularity," in *Proceedings of the 13th International World Wide Web Conference (WWW '04)*, 2004.

[34] Cohen, W., "The Enron Email Dataset." `http://www.cs.cmu.edu/~enron/`, 2005.

[35] Cohen, W. W., "Learning Rules that Classify E-Mail," in *Proceedings of the AAAI Spring Symposium on Machine Learning and Information Access*, pp. 18–25, 1996.

[36] Conti, G. and Ahamad, M., "A taxonomy and framework for countering denial of information attacks," *IEEE Security & Privacy*, vol. 3, no. 6, 2005.

[37] Cormack, G. and Lynam, T., "A Study of Supervised Spam Detection Applied to Eight Months of Personal Email." `http://plg.uwaterloo.ca/~gvcormac/spamcormack.html`, 2004.

[38] Cormack, G. V. and Bratko, A., "Batch and On-line Spam Filter Comparison," in *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS)*, 2006.

[39] Cormack, G. V. and Lynam, T. R., "TREC 2005 Public Spam Corpus." `http://plg.uwaterloo.ca/~gvcormac/treccorpus/`, 2005.

[40] Cormack, G. V. and Lynam, T. R., "TREC 2006 Public Spam Corpus." `http://plg.uwaterloo.ca/~gvcormac/treccorpus06/`, 2006.

[41] Daelemans, W. and others, "TiMBL: Tilburg Memory Based Learner, version 2.0, Reference Guide." `http:/ilk.kub.nl/~ilk/papers/ilk9901.ps.gz`, 1999.

[42] Dalvi, N. and others, "Adversarial classification," in *Proc. 10th Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2004)*, pp. 99–108, 2004.

[43] Davison, B. D., "Recognizing nepotistic links on the web," in *Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search*, 2000.

[44] Donath, J. and Boyd, D., "Public displays of connection," *BT Technology Journal*, vol. 22, no. 4, pp. 71 – 82, 2004.

[45] Drost, I. and Scheffer, T., "Thwarting the nigritude ultramarine: Learning to identify link spam," in *Proceedings of the 16th European Conference on Machine Learning (ECML '05)*, 2005.

[46] Drucker, H., Wu, D., and Vapnik, V., "Support Vector Machines for Spam Categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[47] Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*. Wiley, 1973.

[48] Dumais, S., Platt, J., Heckerman, D., and Sahami, M., "Inductive Learning Algorithms and Representations for Text Categorization," in *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM '98)*, pp. 148–155, 1998.

[49] Ellison, N., Steinfield, C., and Lampe, C., "Spatially bounded online social networks and social capital: The role of facebook," in *Proceedings of the Annual Conference of the International Communication Association (ICA '06)*, 2006.

[50] Fawcett, T., ""In vivo" spam filtering: a challenge problem for KDD," *SIGKDD Explorations Newsletter*, vol. 5, no. 2, pp. 140–148, 2003.

[51] Fawcett, T., "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, 2006.

[52] Fetterly, D. and others, "A large-scale study of the evolution of web pages," in *Proceedings of the 12th International World Wide Web Conference (WWW '03)*, pp. 669–678, 2003.

[53] Fetterly, D., Manasse, M., and Najork, M., "On the evolution of clusters of near-duplicate web pages," in *Proceedings of the 1st Latin American Web Congress*, pp. 37–45, 2003.

[54] Fetterly, D., Manasse, M., and Najork, M., "Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages," in *Proceedings of the 7th International Workshop on the Web and Databases (WebDB '04)*, pp. 1–6, 2004.

[55] Fetterly, D., Manasse, M., and Najork, M., "Detecting phrase-level duplication on the world wide web," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 170–177, 2005.

[56] Fielding, R. and others, "RFC 2616 - hypertext transfer protocol – http/1.1." `http://faqs.org/rfcs/rfc2616.html`, 1999.

[57] Forman, G., "An extensive empirical study of feature selection metrics for text classification," *The Journal of Machine Learning Research*, vol. 3, 2003.

[58] Freed, N. and Borenstein, N., "RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One." http://www.ietf.org/rfc/rfc2045.txt, 1996.

[59] Freiert, M., "February Top Social Networks - Make way for the new guys." http://blog.compete.com/2008/03/07/top-social-networks-traffic-feb-2008%, 2008.

[60] Freund, Y. and Schapire, R. E., "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning*, pp. 148–156, 1996.

[61] Friedman, J., Hastie, T., and Tibshirani, R., "Additive Logistic Regression: A Statistical View of Boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–374, 2000.

[62] Geer, D., "Malicious bots threaten network security," *IEEE Computer*, vol. 38, no. 1, pp. 18–20, 2005.

[63] Golder, S. A., Wilkinson, D., and Huberman, B. A., "Rhythms of social interaction: Messaging within a massive online network," in *Proceedings of the 3rd International Conference on Communities and Technologies (CT '07)*, 2007.

[64] Goodman, J., "Spam Filtering: From the Lab to the Real World." MIT Spam Conference, 2003.

[65] Goodman, J. and Yih, W., "Online Discriminative Spam Filter Training," in *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS)*, 2006.

[66] Google, "Google Directory." http://dir.google.com/, 2005.

[67] Google Press Center, "Google To Acquire YouTube for $1.65 Billion in Stock." http://www.google.com/press/pressrel/google_youtube.html, 2006.

[68] Graham, P., "A Plan for Spam." http://www.paulgraham.com/spam.html, 2002.

[69] Graham, P., "Better Bayesian Filtering." http://www.paulgraham.com/better.html, 2003.

[70] Graham-Cumming, J., "How to Beat a Bayesian Spam Filter." MIT Spam Conference, 2004.

[71] Graham-Cumming, J., "POPFile - Automatic Email Classification." http://popfile.sourceforge.net/, 2004.

[72] Gross, R. and Acquisti, A., "Information revelation and privacy in online social networks," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pp. 71 – 80, 2005.

[73] Gyöngyi, Z. and Garcia-Molina, H., "Link Spam Alliances," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, 2005.

[74] Gyöngyi, Z. and Garcia-Molina, H., "Spam: It's not just for inboxes anymore," *Computer*, vol. 38, no. 10, 2005.

[75] Gyöngyi, Z. and Garcia-Molina, H., "Web spam taxonomy," in *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, 2005.

[76] Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J., "Combating web spam with trustrank," in *Proceedings of the 30th International Conference on Very Large Databases (VLDB '04)*, 2004.

[77] Harris, E., "The Next Step in the Spam Control War: Greylisting." `http://projects.puremagic.com/greylisting/`, 2003.

[78] Henzinger, M. R., Motwani, R., and Silverstein, C., "Challenges in web search engines," *SIGIR Forum*, vol. 36, no. 2, pp. 11–22, 2002.

[79] Heymann, P., Koutrika, G., and Garcia-Molina, H., "Fighting spam on social web sites: A survey of approaches and future challenges," *IEEE Internet Computing*, vol. 11, no. 6, pp. 36 – 45, 2007.

[80] Hidalgo, J. G., "Evaluating Cost-sensitive Unsolicited Bulk Email Categorization," in *Proceedings of the 17th ACM Symposium on Applied Computing*, pp. 615–620, 2002.

[81] Hinduja, S. and Patchin, J. W., "Personal information of adolescents on the internet: A quantitative content analysis of myspace," *Journal of Adolescence*, vol. 31, no. 1, pp. 125 – 146, 2008.

[82] Hirai, J., "Webbase : A repository of web pages," in *Proc. of WWW '00*, 2000.

[83] Hitwise, "Hitwise US - Top 20 Websites - February, 2008." `http://www.hitwise.com/datacenter/rankings.php`, 2008.

[84] Hovold, J., "Naive Bayes Spam Filtering Using Word-Position-Based Attributes," in *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, 2005.

[85] Hulten, G. and others, "Trends in Spam Products and Methods," in *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS '04)*, 2004.

[86] i-config, "Internet Content Filtering Group." `http://iit.demokritos.gr/skel/i-config/`, 2000.

[87] i-config, "Ling-spam." `http://iit.demokritos.gr/skel/i-config/downloads/`, 2000.

[88] Iwanaga, M., Tabata, T., and Sakurai, K., "Evaluation of Anti-Spam Methods Combining Bayesian Filtering and Strong Challenge and Response," in *Proceedings of the IASTED International Conference on Communication, Network, and Information Security (CNIS '03)*, pp. 214–219, 2003.

[89] Jagatic, T. and others, "Social phishing," *Communications of the ACM, to appear*, 2007.

[90] JLCom Publishing Co., "Senate Unanimously Approves Creation of Do-Not-Spam List." `http://www.lawpublish.com/spam.html`, 2003.

[91] Joachims, T., "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization," in *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*, 1997.

[92] Joachims, T., "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Proceedings of the 10th European Conference on Machine Learning (ECML 98)*, pp. 137–142, 1998.

[93] Kidd, E., "Bayesian Whitelisting: Finding the Good Mail Among the Spam." `http://www.randomhacks.net/stories/bayesian-whitelisting.html`, 2002.

[94] King, R., "Marketing to kids where they live." `http://www.businessweek.com/technology/content/sep2006/tc20060908_97440%0.htm?campaign_id=bier_tcs.g3a.091106a`, 2006.

[95] Klimt, B. and Yang, Y., "Introducing the Enron Corpus," in *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS '04)*, 2004.

[96] Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 1995 International Joint Conferences on Artificial Intelligence (IJCAI '95)*, 1995.

[97] Kraut, R. E. and others, "Markets for Attention: Will Postage for Email Help?," in *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, pp. 206–215, 2002.

[98] Krebs, B., "Hacked ad seen on myspace served spyware to a million." `http://blog.washingtonpost.com/securityfix/2006/07/myspace_ad_served_ad%ware_to_mo.html`, 2006.

[99] Kreibich, C. and Crowcroft, J., "Honeycomb: Creating intrusion detection signatures using honeypots," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 51 – 56, 2004.

[100] Kumar, R., Novak, J., and Tomkins, A., "Structure and evolution of online social networks," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, 2006.

[101] Lamb, M., "TarProxy: Lessons Learned and What's Ahead." MIT Spam Conference, 2004.

[102] Lampe, C., Ellison, N., and Steinfield, C., "A familiar face(book): Profile elements as signals in an online social network," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 435 – 444, 2007.

[103] Lewis, D. D. and others, "RCV1: A New Benchmark Collection for Text Categorization Research," *The Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[104] Lowd, D. and Meek, C., "Good Word Attacks on Statistical Spam Filters," in *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS '05)*, 2005.

[105] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B., "Building a Large Annotated Corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[106] McCallum, A. and Nigam, K., "A Comparison of Event Models for Naive Bayes Text Classification," in *Proceedings of the AAAI Workshop on Learning for Text Categorization*, pp. 41–48, 1998.

[107] Messaging Anti-Abuse Working Group, "MAAWG 2006 First Quarter Report." `http://www.maawg.org/about/FINAL\_1Q2006\_Metrics\_Report.pdf`, 2006.

[108] Metsis, V., Androutsopoulos, I., and Paliouras, G., "Spam Filtering with Naive Bayes - Which Naive Bayes?," in *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS)*, 2006.

[109] Microsoft Corporation, "Exchange Intelligent Message Filter." `http://www.microsoft.com/exchange/downloads/2003/imf/default.mspx`, 2003.

[110] Mitchell, T., *Machine Learning*. McGraw Hill, 1997.

[111] Moshchuk, A. and others, "A crawler-based study of spyware in the web," in *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS '06)*, 2006.

[112] Moshchuk, A. and others, "Spyproxy: Execution-based detection of malicious web content," in *Proceedings of the 16th USENIX Security Symposium (Security '07)*, 2007.

[113] Newman, D. J., Hettich, S., Blake, C. L., and Merz, C. J., "UCI Repository of machine learning databases." `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.

[114] NIST, "Text REtrieval Conference (TREC) English Documents." `http://trec.nist.gov/data/docs_eng.html`, 2001.

[115] Ntoulas, A. and others, "Detecting spam web pages through content analysis," in *Proceedings of the 15th International World Wide Web Conference (WWW '06)*, pp. 83–92, 2006.

[116] Ollmann, G., "The Phishing Guide: Understanding & Preventing Phishing Attacks." `http://www.ngssoftware.com/papers/NISR-WP-Phishing.pdf`, 2004.

[117] Pantel, P. and Lin, D., "SpamCop: A Spam Classification & Organization Program," in *Proceedings of the AAAI Workshop on Learning for Text Categorization*, 1998.

[118] PCWorld.com, "California Anti-Spam Law." `http://www.pcworld.com/downloads/file_description/0,fid,23113,tfg,tfg,0%0.asp`, 2006.

[119] Prakash, V. V., "Vipul's Razor." `http://razor.sourceforge.net/`, 2006.

[120] Prince, M. and others, "Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honey pot," in *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS '05)*, 2005.

[121] Provos, N. and others, "The ghost in the browser: Analysis of web-based malware," in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots '07)*, 2007.

[122] Qiu, F., Liu, Z., and Cho, J., "Analysis of user web traffic with a focus on search activities," in *Proceedings of the 8th International Workshop on the Web and Databases (WebDB '05)*, 2005.

[123] Quinlan, J. R., *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[124] Rabin, M., "Fingerprinting by random polynomials," Tech. Rep. TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[125] Raudys, S. J. and Jain, A. K., "Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, 1991.

[126] Ross, B. and others, "Stronger password authentication using browser extensions," in *Proceedings of the 14th Usenix Security Symposium*, pp. 17–32, 2005.

[127] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E., "A Bayesian Approach to Filtering Junk E-Mail," in *Proceedings of the AAAI Workshop on Learning for Text Categorization*, pp. 55–62, 1998.

[128] Sakkis, G. and others, "Stacking Classifiers for Anti-spam Filtering of E-mail," in *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, pp. 44–50, 2001.

[129] Salton, G., Wong, A., and Yang, C. S., "A Vector Space Model for Automatic Indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[130] Samy, "Technical explanation of the MySpace worm." `http://namb.la/popular/tech.html`, 2005.

[131] Sanchez, M., "Pranksters posting fake profiles on myspace." `http://www.dfw.com/mld/dfw/news/local/15255785.htm?template=contentModu%les/printstory.jsp`, 2006.

[132] Schapire, R. E. and Singer, Y., "Improved Boosting Algorithms Using Confidence-rated Predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[133] Schapire, R. E., Singer, Y., and Singhal, A., "Boosting and Rocchio Applied to Text Filtering," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 215–223, 1998.

[134] Schneider, K., "A Comparison of Event Models for Naive Bayes Anti-Spam E-mail Filtering," in *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL '03)*, pp. 307–314, 2003.

[135] Schneider, K., "Brightmail URL Filtering." MIT Spam Conference, 2004.

[136] Seibel, J., "Boy charged in creating fake Myspace profile." `http://www.jsonline.com/story/index.aspx?id=413620`, 2006.

[137] SiteAdvisor, "Protection from spyware, spam, viruses and online scams — siteadvisor." http://www.siteadvisor.com/, 2006.

[138] Sophos, "Sophos identifies the most prevalent spam categories of 2005." http://www.sophos.com/pressoffice/news/articles/2005/08/pr_us_20050803t%opfive-cats.html, 2005.

[139] SpamArchive, "SpamArchive.org - Donate your Spam to Science." http://www.spamarchive.org/, 2002.

[140] SpamAssassin Development Team, "The Apache SpamAssassin Project." http://spamassassin.apache.org/, 2001.

[141] SpamAssassin Development Team, "The SpamAssassin Public Corpora." http://spamassassin.apache.org/publiccorpus/, 2004.

[142] Spitzner, L., "The honeynet project: Trapping the hackers," *IEEE Security & Privacy*, vol. 1, no. 2, pp. 15 – 23, 2003.

[143] Swartz, J., "Social-networking sites going global." http://www.usatoday.com/money/industries/technology/2008-02-10-social-n%etworking-global_N.htm, 2008.

[144] Technocrat, "Myspace phishing attacks on the rise." http://djtechnocrat.blogspot.com/2006/05/myspace-phishing-attacks-on-ri%se.html, 2006.

[145] Templeton, B., "E-Stamps." http://www.templetons.com/brad/spam/estamps.html, 2003.

[146] The Linguist List, "The Linguist List." http://listserv.linguistlist.org/archives/linguist.html, 1990.

[147] Vapnik, V. N., *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[148] Wade, R., "Fakesters: On myspace, you can be friends with burger king. this is social networking?." http://www.technologyreview.com/Infotech/17713/, 2006.

[149] Wang, Y. and others, "Strider typo-patrol: Discovery and analysis of systematic typo-squatting," in *Proceedings of the 2nd Workshop on Steps to Reduce Unwanted Traffic on the Internet (SRUTI '06)*, pp. 31–36, 2006.

[150] Wang, Y. and others, "Spam double funnel: Connecting web spammers with advertisers," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, 2007.

[151] Wittel, G. L. and Wu, S. F., "On Attacking Statistical Spam Filters," in *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, 2004.

[152] Witten, I. H. and Frank, E., *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, 2000.

[153] Wu, B. and Davison, B. D., "Cloaking and redirection: A preliminary study," in *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, 2005.

[154] Wu, B. and Davison, B. D., "Identifying link farm spam pages," in *Proceedings of the 14th International World Wide Web Conference (WWW '05)*, 2005.

[155] Yang, Y. and Pederson, J. O., "A comparative study of feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*, pp. 412–420, 1997.

[156] Zhang, L., Zhu, J., and Yao, T., "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 4, pp. 243–269, 2004.

[157] Zinman, A. and Donath, J., "Is britney spears spam?," in *Proceedings of the 4th Conference on Email and Anti-Spam (CEAS '07)*, 2007.

VITA

Steve Ross Webb majored in Computer Science at Baylor University and ultimately graduated summa cum laude with a 4.0 grade point average. It was at Baylor that Steve discovered his passion for computing research, and based on the encouragement of his Honors Program thesis advisor, Dr. Jeff Donahoo, Steve applied to the Ph.D. program at Georgia Tech. In the summer of 2003, Steve began his journey as a Ph.D. student at Georgia Tech under the supervision of Dr. Calton Pu. While at Tech, Steve was affiliated with the Center for Experimental Research in Computer Systems (CERCS), the Georgia Tech Information Security Center (GTISC), and the Distributed Data Intensive Systems Lab (DISL). Steve's primary research project was the Denial of Information (DoI) Project, and his research focused on the removal of low-quality information from online information-rich environments (e.g., email systems, the World Wide Web, social environments, etc.).