Predicting Web Spam with HTTP Session Information

Steve Webb
College of Computing
Georgia Tech
Atlanta, GA 30332
webb@cc.gatech.edu

James Caverlee
Dept. of Computer Science
Texas A&M University
College Station, TX 77843
caverlee@cs.tamu.edu

Calton Pu
College of Computing
Georgia Tech
Atlanta, GA 30332
calton@cc.gatech.edu

ABSTRACT

Web spam is a widely-recognized threat to the quality and security of the Web. Web spam pages pollute search engine indexes, burden Web crawlers and Web mining services, and expose users to dangerous Web-borne malware. 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. In this paper, 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.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; I.5.2 [**Pattern Recognition**]: Design Methodology—*classifier design and evaluation*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Algorithms, Experimentation, Measurement, Security

Keywords

Web spam, HTTP session information, classification

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-59593-991-3/08/10 ...$5.00.

1. INTRODUCTION

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 [4, 22]) as well as pages designed to propagate malware by hosting Web-borne spyware and browser exploits [20, 21, 23, 25]. 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 [2, 3, 5, 6, 7, 16, 29] and content-based [11, 22, 27] 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 paper 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 our previous work [27], which shows that very distinct patterns emerge within the HTTP session information associated with Web spam pages. In this paper, 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 paper 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 paper 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 $101\mu s$ 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 paper is organized as follows. Section 2 reviews previous research on Web spam identification and classification. Section 3 provides background information about HTTP operations and describes our new approach for retrieving Web pages. Section 4 describes the experimental setup we used to evaluate our new approach, and Sections 5 and 6 present our experimental results using various corpora of spam and legitimate Web pages. We conclude the paper in Section 7.

2. 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 [8] was the first to investigate link-based Web spam, building decision trees to successfully identify "nepotistic links." Becchetti et al. [2] 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. [16] 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 [29] and Benczur et al. [3] 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, in our previous research, we proposed a spam resilient ranking approach called Spam-Resilient SourceRank [7]. 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. Additionally, we presented a credibility-based Web ranking algorithm called CredibleRank [6]. This algorithm assesses the link credibility of Web pages, and then, it incorporates that credibility information into the rankings of each page on the Web.

Content-based analysis techniques focus on identifying fundamental differences between the content of spam and legitimate Web pages. Fetterly et al. [11] 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. [22] 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. [1] 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. Finally, in our previous work [27], we performed a large-scale characterization of 348,878 Web spam pages, which focused on various content-based components of Web spam (e.g., content duplication, content categorization, redirection, etc.).

Hybrid analysis techniques combine content-based and link-based techniques to identify Web spam. Gyöngyi and Garcia-Molina [15] 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 and Scheffer [9] 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%. Finally, Castillo et al. [5] created a cost-sensitive decision tree using the content-based features described in [22] and link-based features described in [2]. Their decision tree was able to successfully identify 88.4% of their 675 Web spam examples with a false positive rate of 6.3%.

3. 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.

3.1 Traditional Approach

The HyperText Transfer Protocol (HTTP) [12] 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.

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

Figure 1: Example HTTP request message.

The information exchange between user agents and origin servers is accomplished by transmitting MIME-like messages (as specified in RFC 2616 [12]) back and forth between 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 1.

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 2 shows the annotated response message that corresponds to the "GET" request message shown in Figure 1. 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").

```
 Status Line | 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
     Headers | 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 |
             | <!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" />
Message Body |   <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 2: Example HTTP response message.

3.2 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 3.1. 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 [20, 23, 25]. 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.

4. 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 5 and 6, we provide experimental evidence that HTTP session classification is an effective and efficient solution for automatically predicting Web spam pages.

4.1 Classification Framework

Previous research [2, 5, 8, 9, 22] 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 [18]. 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.

4.2 Corpora

In our previous research [26], 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[1]. This corpus is currently the largest publicly available collection of Web spam, and it served as our primary source of spam instances. For a more detailed description of our collection methodology and the format of the files in the Webb Spam Corpus, please consult [26].

In addition to the Webb Spam Corpus, we also used the labeled spam instances in the WEBSPAM-UK2006 corpus[2] as a source of Web spam pages. The WEBSPAM-UK2006 corpus [4] 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 [2, 5], 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.

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

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

Table 1: 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

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 [17]. 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.

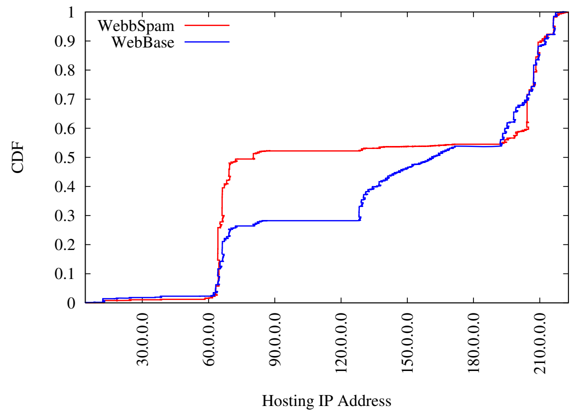
In addition to our WebBase data, we also used the labeled legitimate instances in the WEBSPAM-UK2006 corpus as a source of legitimate Web data. As mentioned above, this corpus does not contain HTTP session information; thus, we recrawled the legitimate URLs from the corpus to obtain that data. We performed 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 missing pages were no longer available.

Table 1 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.

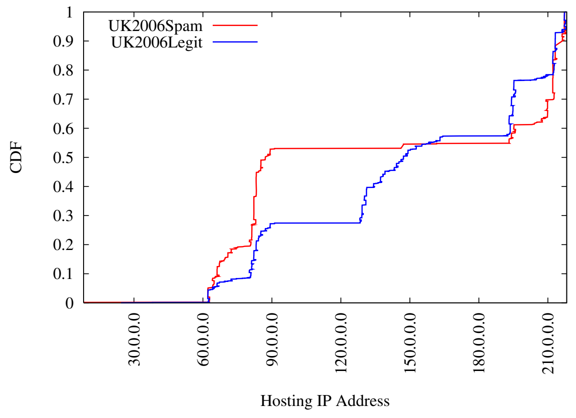
4.3 HTTP Session Information Features

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 [24] (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 infor-



(a) WebbSpam and WebBase



(b) UK2006Spam and UK2006Legit

Figure 3: 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.

Table 2: 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)	Sample Value
Content-Type	348,878 (688)	392,600 (122)	1,338 (41)	3,542 (123)	text/html
Server	343,168 (6,513)	387,301 (2,505)	1,336 (196)	3,490 (704)	apache/2.0.52 (fedora)
Connection	327,478 (6)	354,837 (8)	1,322 (2)	3,230 (4)	close
X-Powered-By	209,215 (261)	114,181 (111)	862 (52)	1,753 (90)	asp.net
Content-Length	162,532 (31,232)	214,100 (58,417)	498 (447)	2,203 (1,987)	1470
Cache-Control	148,715 (548)	103,916 (1,461)	283 (27)	1,377 (117)	private
Set-Cookie	145,315 (140,431)	110,519 (105,636)	287 (287)	1,392 (1,376)	gx_jst=9fa7274e662d6164; path=/apps/system
Link	142,785 (15,573)	79 (64)	797 (421)	2,690 (2,215)	<style.css>; rel="stylesheet"; type="text/css"
Expires	93,477 (25,056)	55,991 (31,197)	183 (44)	745 (353)	mon, 26 jul 1997 05:00:00 gmt
Pragma	75,435 (32)	29,464 (9)	167 (5)	541 (10)	no-cache
Last-Modi↓ed	73,071 (50,206)	149,191 (125,573)	418 (380)	1,348 (1,245)	wed, 21 dec 2005 00:21:06 gmt
P3P	59,023 (816)	21,970 (205)	27 (14)	84 (54)	cp="noi cura adma our nor bus phy onl uni pur com nav sta"
Accept-Ranges	47,821 (4)	157,384 (4)	442 (1)	1,245 (2)	bytes
X-Meta-Robots	45,900 (241)	N/A	327 (25)	662 (64)	index, follow
Refresh	45,075 (8,556)	136 (6)	9 (8)	150 (137)	0; url=/index.asp
Etag	42,546 (24,783)	118,804 (115,122)	347 (317)	1,125 (1,039)	"110d66-112-24dfc0"
Vary	25,721 (45)	15,939 (27)	43 (5)	120 (12)	accept-encoding, user-agent
Content-Language	20,397 (163)	9,661 (49)	138 (7)	401 (18)	en-us

mation. 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 3(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, 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 3(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 2 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. The table also presents a sample value for each of the headers. 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 2 is that distinct distributions exist for the HTTP session header values

Table 3: 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

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 3 shows the features that were derived from the sample "Server" header shown in Table 2 ("apache/2.0.52 (fedora)").

4.4 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 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 [13, 30] 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.

4.5 Classifiers

Unlike previous Web spam classification research [2, 5, 8, 22], 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 [28]. 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 paper. Additional information about the classifiers is available in most standard machine learning texts (e.g., [19]).

5. 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.

5.1 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 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 4) and a corpus sample consisting of 10,000 data instances. Therefore, those settings were used

Table 4: 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=

Table 5: Classifier performance results for Webb Spam and WebBase.

Classifier	TP	FP	F-Measure	Accuracy
Content-based [22]	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%

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 4.5. 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 5.2 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 5. 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 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 [10]. Figure 4 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.

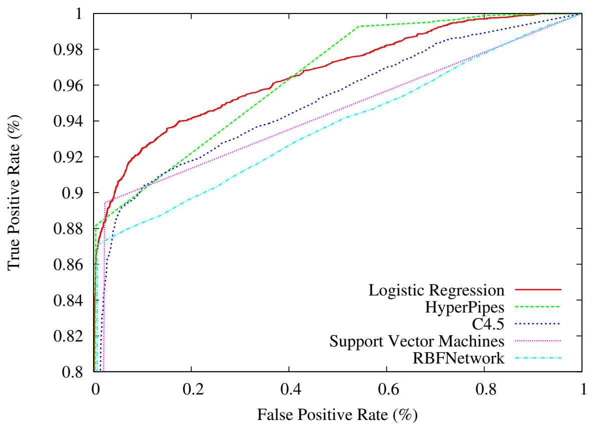


Figure 4: ROC curves for the top 5 classifiers using Webb Spam and WebBase.

To help put our results in their proper context, Table 5 also shows the best results from a previously studied content-based Web spam classifier [22]. 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 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.

5.2 Class Distribution Evaluation

Based on the empirical growth patterns exhibited by Web spam [14], 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 6. As the table shows, the

Table 6: 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%

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 are all within about 10% of their best values.

5.3 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 5.1. 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 7. 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 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 overhead will be completely unnoticeable by Web users, and it should also have almost no effect on the operations of Web crawlers.

5.4 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.

Table 7: 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

Table 8: 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

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.

6. WEBSPAM-UK2006 RESULTS

In Section 5, 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 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.

6.1 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 5.1. 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 8). Due to the limited size of this corpus, we did not evaluate the impact of corpus sample size on the classifiers.

Table 9: Classifier performance results for WEBSPAM-UK2006.

Classifier	TP	FP	F-Measure	Accuracy
Hybrid [5]	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%

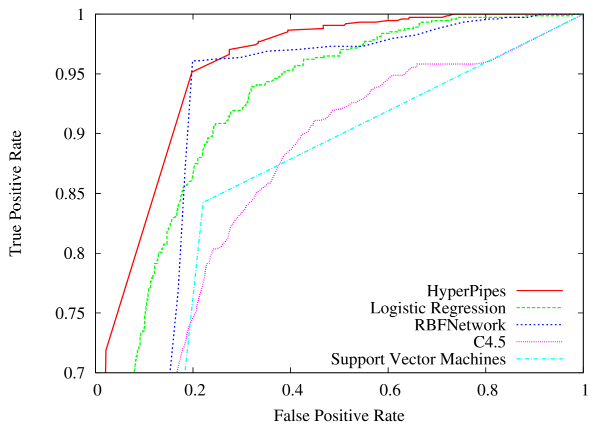


Figure 5: ROC curves for the top 5 classifiers using WEBSPAM-UK2006.

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 5.2. 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., 743 instances of each class) to evaluate the performance of the 40 classification algorithms described in Section 4.5. Table 9 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 5, and the FP rates for all of the classifiers increased. A similar performance degradation is also evident in the ROC graph shown in Figure 5.

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 5.1. Unfortunately, this was unavoidable due to the limited size of the WEBSPAM-UK2006 corpus. Second, previous research [5] 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 9 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) [5]. 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 significantly lower FP rate, which prompted us to use it for the remaining evaluations.

Table 10: 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%

6.2 Class Distribution Evaluation

As mentioned above in Section 5.2, 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 5.2, 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 10. The table shows somewhat erratic TP rates, FP rates, and F-Measure values as the percentage of spam increases. Unlike Table 6, which shows a slow decline for these metrics as the spam percentage increases, the results in Table 10 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 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 5.2, we believe these performance fluctuations would be reduced.

Table 11: 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

6.3 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 6.1. Using the same approach described in Section 5.3, we computed the training time and per instance classification time necessary for each classifier to perform a stratified tenfold cross-validation evaluation. Table 11 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 7. However, the training times in Table 11 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 11. As a result, we believe the timings obtained on the larger corpora in Section 5.3 are more indicative of our approach's efficiency.

6.4 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.

7. CONCLUSIONS AND FUTURE WORK

In this paper, 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.

Our predictive approach is complementary to previous Web spam research focused on link-based and content-based analysis techniques. 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.

8. REFERENCES

[1] D. S. Anderson et al. Spamscatter: Characterizing internet scam hosting infrastructure. In *Proc. of Usenix Security '07*, 2007.

[2] L. Becchetti et al. Link-based characterization and detection of web spam. In *Proc. of AIRWeb '06*, 2006.

[3] A. A. Benczur et al. Spamrank - fully automatic link spam detection. In *Proc. of AIRWeb '05*, 2005.

[4] C. Castillo et al. A reference collection for web spam. *SIGIR Forum*, 40(2), 2006.

[5] C. Castillo et al. Know your neighbors: Web spam detection using the web topology. In *Proc. of SIGIR '07*, 2007.

[6] J. Caverlee and L. Liu. Countering web spam with credibility-based link analysis. In *Proc. of PODC '07*, 2007.

[7] J. Caverlee, S. Webb, and L. Liu. Spam-resilient web rankings via influence throttling. In *Proc. of IPDPS '07*, 2007.

[8] B. D. Davison. Recognizing nepotistic links on the web. In *Proc. of AIWS '00*, 2000.

[9] I. Drost and T. Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proc. of ECML '05*, 2005.

[10] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 2006.

[11] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *Proc. of WebDB '04*, 2004.

[12] R. Fielding et al. RFC 2616 - hypertext transfer protocol – http/1.1. `http://faqs.org/rfcs/rfc2616.html`, 1999.

[13] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3, 2003.

[14] Z. Gyöngyi and H. Garcia-Molina. Spam: It's not just for inboxes anymore. *Computer*, 38(10), 2005.

[15] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proc. of AIRWeb '05*, 2005.

[16] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proc. of VLDB '04*, 2004.

[17] J. Hirai. Webbase : A repository of web pages. In *Proc. of WWW '00*, 2000.

[18] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of IJCAI '95*, 1995.

[19] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[20] A. Moshchuk et al. A crawler-based study of spyware in the web. In *Proc. of NDSS '06*, 2006.

[21] A. Moshchuk et al. Spyproxy: Execution-based detection of malicious web content. In *Proc. of Usenix Security '07*, 2007.

[22] A. Ntoulas et al. Detecting spam web pages through content analysis. In *Proc. of WWW '06*, 2006.

[23] N. Provos et al. The ghost in the browser: Analysis of web-based malware. In *Proc. of HotBots '07*, 2007.

[24] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Comm. of the ACM*, 18(11), 1975.

[25] Y. M. Wang et al. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *Proc. of NDSS '06*, 2006.

[26] S. Webb, J. Caverlee, and C. Pu. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *Proc. of CEAS '06*, 2006.

[27] S. Webb, J. Caverlee, and C. Pu. Characterizing web spam using content and http session analysis. In *Proc. of CEAS '07*, 2007.

[28] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[29] B. Wu and B. D. Davison. Identifying link farm spam pages. In *Proc. of WWW '05*, 2005.

[30] Y. Yang and J. O. Pederson. A comparative study of feature selection in text categorization. In *Proc. of ICML '97*, 1997.