Improving Web Spam Classification using Rank-time Features

Krysta M. Svore
Microsoft Research
1 Microsoft Way
Redmond, WA
ksvore@microsoft.com

Qiang Wu
Microsoft Research
1 Microsoft Way
Redmond, WA
qiangwu@microsoft.com

Chris J.C. Burges
Microsoft Research
1 Microsoft Way
Redmond, WA
cburges@microsoft.com

Aaswath Raman
Microsoft
1 Microsoft Way
Redmond, WA
aaswathr@microsoft.com

ABSTRACT

In this paper, we study the classification of web spam. Web spam refers to pages that use techniques to mislead search engines into assigning them higher rank, thus increasing their site traffic. Our contributions are two fold. First, we find that the method of dataset construction is crucial for accurate spam classification and we note that this problem occurs generally in learning problems and can be hard to detect. In particular, we find that ensuring no overlapping domains between test and training sets is necessary to accurately test a web spam classifier. In our case, classification performance can differ by as much as 40% in precision when using non-domain-separated data. Second, we show rank-time features can improve the performance of a web spam classifier. Our paper is the first to investigate the use of rank-time features, and in particular query-dependent rank-time features, for web spam detection. We show that the use of rank-time and query-dependent features can lead to an increase in accuracy over a classifier trained using page-based content only.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia; K.4.m [**Computers and Society**]: Miscellaneous; H.4.m [**Information Systems**]: Miscellaneous

General Terms

Measurement, Experimentation, Algorithms

AIRWeb '07, May 8, 2007 Banff, Alberta, Canada.

Keywords

Web characterization, web pages, web spam, data mining

1. INTRODUCTION

As the amount of information on the World Wide Web grows, the use of search engines to find relevant information becomes ever more critical. A search engine retrieves pages relevant to a user's query by comparing the attributes of pages, together with other features such as anchor text, and returning those that best match the query. The user is then typically shown a ranked list of between 10 and 20 URLss per page. The ranking of pages by search engines has proved to be a crucial component of how users browse the web. This arises not only from the simple gathering of information for users, but also from commercial transactions that result from the search activity.

Some commercial companies, to increase their website traffic, hire *search engine optimization* (SEO) companies to improve their site's ranking. There are numerous ways to improve a site's ranking, which may be broadly categorized as ethical, or white-hat, and less ethical, or gray-hat (or black-hat), SEO techniques. White-hat SEO methods focus on improving the quality and content of a page so that the information on the page is useful for many users. Such a method for improving a site's rank may be to improve the content of the site, such that it appears most relevant for those queries one would like to target.

However, there are many unethical methods of improving ranking. Gray-hat and black-hat SEO techniques include methods such as *link stuffing*, *keyword stuffing*, *cloaking*, *web farming*, and so on [8]. Link stuffing is the practice of creating many pages that have little content or duplicated content, all of which link to a single optimized target page. Link-stuffed pages can fool a ranker based on link structure into thinking that the target page is a better page since so many pages link to it. Keyword stuffing is when a page is filled with query terms, making it appear very relevant to a search whose query contains one or more of those terms, even though the actual relevance of the page may be low. A keyword-stuffed page will rank higher since it appears to have content relevant to the query. For overloaded terms on the page, the page will appear higher in the search results

and will ultimately draw users to click on the site. For a more detailed list of gray-hat techniques, see [8].

Web spam is defined as a page that uses gray-hat or black-hat methods to improve ranking; it is designed solely to increase ranking and not to increase the quality or content of the page[1]. Typically, webspam pages are either useless to a user or simply do not reveal as high of quality of content as an actual relevant page.

As a result, search engines now face the challenge of eliminating or demoting these types of pages in the ranking of results. This challenge is not simple to overcome since the webspam pages' content may be duplicated from a quality page, the link structure may be complicated to detect, or they may hide pages' content or functionality from the web crawler through cloaking techniques [14]. Therefore, it is hard to determine which page features are the best to differentiate between web spam and relevant web pages.

The elimination of web spam from search results is important for many reasons. First, users who consistently see spam ranked highly in search results will over time turn to another search engine to perform information retrieval. Second, significant financial gains can occur due to the position of a site in the search engine ratings. If spam sites are consistently in the top positions, then ultimately legitimate sites are deprived of those financial gains. This can lead to good sites using spamming techniques to improve their ratings. Third, if there are many spam pages in a search engine's index, then the search engine wastes resources on illegitimate sites. Each page must be crawled, indexed, and searched for each query to the search engine. If a significant fraction of the indexed pages are in fact spam, the search engine (and the end users) lose in many ways. Therefore, it is crucial to devise automated spam detection methods. Human labeling alone cannot scale to handle the size of the problem.

Currently, web search engines employ an array of spam detectors targeted toward different kinds of web spam. Spam can potentially be caught at many levels: at crawl-time, at index-generation time, or at rank time. Of course, it is best to remove spam as early in the chain as possible, so as not to waste valuable resources on web spam pages. Ideally, we desire an index *free* of spam. However, such a scenario is virtually impossible. Thus, for our study we focus on pages already in the index and perform spam detection at rank time.

In this paper, we begin by identifying a common practice within the literature, in particular the training and testing of a web spam classifier on a random sampling of URLs, and highlight the inaccuracies in this approach. We demonstrate that classification performance on a random set of URLs is misleading since the classifier may in fact be learning a relationship between feature vectors and domain. We urge the use of domain-separated datasets to determine accurate classification performance. We emphasize that although this problem may seem obvious in hindsight, it does occur in much of the literature.

We then build a classifier to detect web spam by training a linear SVM-based classifier on a set of page attributes. We study the addition of rank-time features to our initial feature set, which is a novel approach to spam detection. The use of rank-time features, of which most are query dependent, may seem counterintuitive at first, since a page is either spam or not spam, regardless of the query used to find it. However, if the spam page has fooled the ranking algorithms, then that act alone may give spam pages properties which are missing from legitimate pages. A good way to appreciate this is to consider ranking algorithms that use feature vectors, derived from the query, the document, and other sources such as anchor text. Those feature vectors have a distribution, and in general it should be difficult for spammers to match this distribution since they do not have access to large amounts of ranking data. For example, a linear neural net classifier [3] with a positive weight for the number of query words that occur in the body of the page will be easily fooled by simply increasing that number; but such a feature vector will be an outlier from the distribution of feature vectors generated from legitimate pages. Our studies show improved classification performance when using rank-time features.

Our paper is organized as follows. In Section 2 we briefly describe support vector machines and the web spam detection problem. In Section 3 we describe the construction of our real-world datasets. We highlight the importance of domain-separated data and present several compelling examples in Section 4. We briefly describe our initial set of features in Section 5 and also examine the accuracy of our classifier based on these features. In Section 6 we motivate the use of rank-time features for classification and describe both our query-independent and query-dependent rank-time features. The results on our datasets using the rank-time features are discussed in Section 7. In Section 8 we discuss related work and finally, in Section 9, we conclude and comment on future directions of our work.

2. WEB SPAM DETECTION

Web spam detection can be viewed as a classification problem. To detect web spam pages, we build a classifier to label a given web page as spam or not spam. Classification consists of training and test data that is composed of a number of labeled instances, or samples, where each sample has a vector of attributes, or features. In our case, the labels are determined by human judges, as described in Section 3.

Classification involves producing a model during training to predict the label of each instance in the test set given only the vector of feature values. To construct a classifer, we first train it on a number of labeled training samples and determine the parameters of our classifier. During testing, the classifier examines a vector of features jointly to determine, based on the feature values, if a web page is spam or not. The classifier is evaluated during testing by comparing the label given by the classifer with the instance's assigned label. Our feature set includes variants of page- and domain-level features. Our feature set will be discussed in Section 5.

There are many different classification algorithms that can be used to detect web spam. In this paper, we demonstrate the use of rank-time features to improve web spam classification using a support vector machine. At a high level, a support vector machine, developed by Vapnik [12], produces a linear separating hyperplane between two class labels in a transformed version of the feature space. Instances are then classified based on where they lie in the transformed version of feature space. The SVM finds the separating hyperplane with maximal margin in the high-dimensional space, where $C > 0$ measures how much to penalize the error term. In this paper, we apply a linear kernel function: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. We choose a linear SVM since it is simple and tends to

[1]This is not to be confused with email spam, which is typically referred to as "spam".

Table 1: Sizes of our datasets.

Dataset	# Total	# Spam	# Non-spam	% Spam
1	6334	705	5629	11.13
2	6291	673	5618	10.70
3	6324	639	5685	10.10
4	6268	614	5654	9.80
5	6083	502	5581	8.25

perform well on high-dimensional problems. Our goal is to demonstrate the performance gain by using rank-time features for classification.

3. DESCRIPTION OF DATASETS

In this paper, we emphasize that we focus on page-level, content-based classification, as opposed to host-level classification or link-level classification. Our rank-time web spam classifier could be used in conjunction with a domain-level or link-level classifier, by using our classifier at rank time and another classifier at index-generation time, for example. We do include, however, features based on domain and link information. Building a supervised web spam classifier requires labeled samples (see [4] for a relevant discussion of a labeling process of another (public) spam dataset). For our experiments, we use a large set of human-labeled (query, URL) pairs. First we decide on a list of queries from the Microsoft Live search engine query logs. The queries are frequency subsampled such that the set of queries we use represent a realistic distribution of queries that users would submit to a search engine. The query frequencies are determined from toolbar data as well as query logs. Queries include commercial queries, spam, queries, and non-commercial queries. Our query set consists of 1697 unique queries.

A human judge is then given the list of queries and issues each query. A returned list of 10 results with snippets is shown to the judge. For each URL appearing in the top 10 returned search results, the judge labels the URL as spam, not spam, or unknown. The judgment is made based on the quality of content, the use of obvious spam techniques, and whether or not the result should appear in the top 10. In our datasets, we use only URLs labeled as spam or non-spam and choose not to use URLs labeled as unknown. The URLs were gathered for our query set in July 2006.

To measure our performance, we perform five-fold cross validation. The five datasets were constructed by randomly assigning each domain in our set of URLs to one of five files and then assigning every URL from that domain to that file. On average, there are 4105 domains per file. Five-fold cross validation consists of training our model on four of the five files and testing the performance of the model on the remaining file. We repeat this for all combinations of files. Our collection contains 31300 total URLs, of which 3133 are labeled spam (9.99% spam). We divide our collection into five similar-sized files, as listed in Table 1.

Note that with only 10% spam on average in our files that the problem of web spam detection becomes very difficult. Previous studies have used datasets with around 25% spam [4, 11], thus previously reported results have higher precision and recall than the results reported in this paper.

4. DOMAIN SEPARATION IN DATASETS

In constructing our five files, we could have randomly assigned each URL to one of five files, but instead we randomly assigned a domain to each file. We find that training and testing on non-domain-separated data can result in misleading performance; this is a critical point in this paper. We found in our studies that separating the data by domain among the training and test sets was crucial. The problem here is in fact a general problem facing any machine learning algorithm when the data contains large-scale structures that may go unnoticed by the researcher. We briefly describe here the problem as we encountered it, and then we briefly explain why we believe that this problem is general, and insidious, in that it can invalidate results in a way that goes unnoticed by those who train machine learning classifiers.

For the problem investigated in this paper, suppose that the separation into the training and test sets was done randomly, that is, the feature vectors were chosen in random order from the data. Since spammers often buy large blocks of domains for their purposes, it is often the case that entire domains are spam[2]. Suppose now that one of the features is a number that is generated by a hash of the domain name. If most domains in the data are either all spam, or all not spam, then the classifier can simply learn the mapping from that hash value to the spam label. If care was not taken to separate the data by domain, then the classifier can appear to do well on the (randomly chosen) test set, whereas in fact it generalizes very poorly on new domain data.

In the instantiation of this problem considered in this paper (separation of the data by domain), domain separation is a fairly obvious precaution to take, although we do take this opportunity to warn other workers of this danger, since even with reasonably large amounts of data, this effect can grossly bias the results. Previous reports in the literature have failed to separate data by domain, thus *performance results in the literature may in fact be worse than reported.*

However this problem can occur in any machine learning classification task. If there exist subsets of the data (analogous to domains for the spam classification problem), for which, in a given subset, most of the data has the same label, and for which the algorithm can more easily learn the mapping from the feature vectors to the subsets, rather than the mapping from the feature vectors to the labels we care about, then measured test results can turn out to be wildly and incorrectly optimistic. In general, this may be far harder to detect than in the spam case considered in this paper: the mapping from input features, to subset hash, may be nonlinear and non-obvious, and the subsets themselves may be objects whose existence is not suspected. By testing on domain-separated data, we obtain worst-case performance results. However, our approach does not solve the training problem; the model still learns a mapping from feature vectors to subset hash to labels instead of feature vectors to labels. We leave this general issue to future work.

4.1 Motivating example

To motivate the need for domain-separated training and test sets, we present compelling results using non-domain-separated and domain-separated datasets. We construct a model using a feature set that includes dynamic-rank features that are both query-independent and query-dependent,

[2]This is not always the case, e.g., for openly editable pages that contain guest links to spam pages.

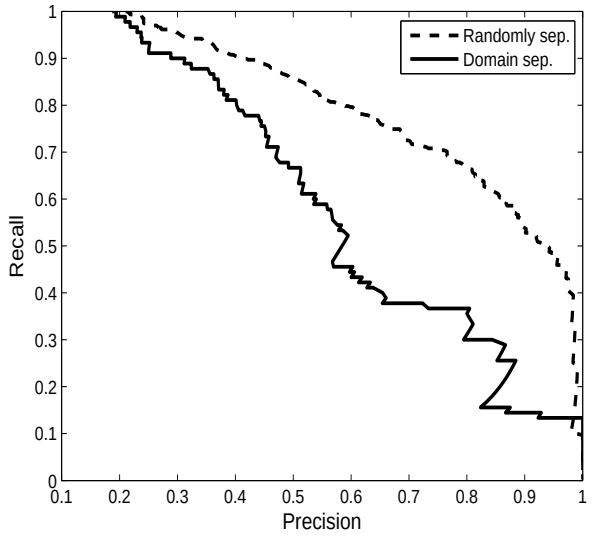

Figure 1: Precision vs. recall on randomly separated (dashed line) and domain-separated (solid line) test sets using the feature set $FEAT_{A+B+Q}$ on test sets consisting of 25% spam. Results on randomly separated data achieve 90% precision at 50% recall. Results on domain-separated data indicate less than 60% precision at 50% recall.

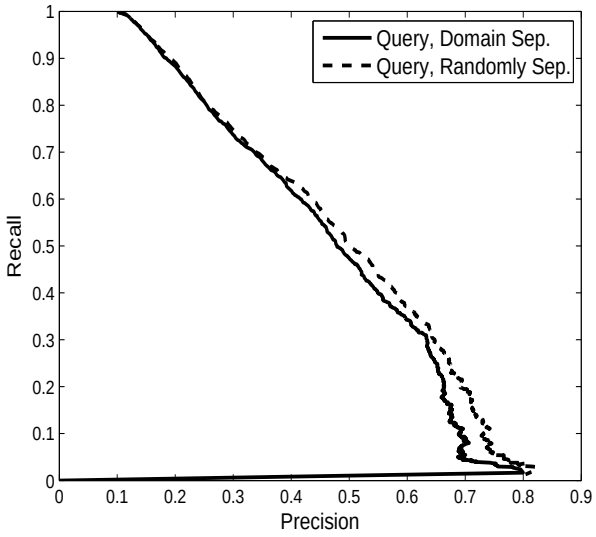

Figure 2: Precision vs. recall on randomly separated (dashed line) and domain-separated (solid line) test sets using the feature set $FEAT_{A+B+Q}$ on domain-separated test sets with 10% spam. Results on randomly separated data indicate improved performance over results on domain-separated data.

denoted by $FEAT_{A+B+Q}$. These features are described in Section 6. We construct two datasets: the first is domain-separated and the second is random. Domain separation means that for a given URL pair, the same domain cannot appear in both the training and test sets.

We construct a linear SVM-based model using the randomly separated datasets. We then repeat our experiment, but now train a model using our domain-separated datsets, using the same set of features. With our particular choice of features, we can achieve almost 100% precision at 50% recall on the randomly separated datasets. Figure 1 shows the precision–recall curve achieved using randomly separated datasets and domain-separated datasets on feature set $FEAT_{A+B+Q}$(see Section 6), for datasets consisting of 25% spam.

Evaluation of our results for the domain-separated datasets indicates that at 50% recall the precision drops to less than 60%. Such a noticeable drop in precision occurs because the model has learned a mapping from feature vectors to domain to label instead of a mapping from feature vectors to label. Performance on our test sets declines since the model cannot rely on the feature vectors to domain mapping it has learned since there are no overlapping domains between training and test sets. In the worst case, there will be no overlapping domains between training and test sets, so our results on domain-separated data are reflective of worst-case performance. We also experimented with removing obviously domain-related features from our feature set and the classifier is still able to learn a mapping from feature vector to domain to label from some other combination of features.

Figure 2 shows the precision–recall curve achieved using the randomly separated datasets on our rank-time-dependent feature set $FEAT_{A+B+Q}$ (see Section 6), for datasets consisting of 10% spam. Table 1 lists the details of our training and test sets used for results shown in Figure 2. Since a dataset consisting of 10% spam is more difficult, the precision drops on both randomly separated and domain-separated test sets. However, testing on randomly separated data still indicates slightly misleading performance; when we separate by domain, the model is unable to abuse the learned feature vectors to domain mapping.

A subtlety such as the learning of a mapping from feature vectors to some subset, such as domain, and then to label can be recognized in this instance since we are able to examine the results and perform two experiments that clearly indicate this learning behavior. However, in other circumstances, it may not be apparent that such "cheating" is occurring. We warn the reader that the presence of such structures may be hard to detect and caution classifier designers against such woeful misfortunes. How to choose a set of features to prevent learning a connection is an open research problem. For the remainder of the paper, we report only on the domain-separated datasets given in Table 1. In this way, we are certain to report worst-case performance results.

5. RANK-INDEPENDENT CLASSIFICATION MODEL

Our initial classification model is built using URL page-level attributes. The attributes include domain-level features, page-level features, and link information. We determine the values of all features by mining feature information for each URL in our test and training sets. We initially build a model using features that take advantage of basic domain

Table 2: Sample features in $FEAT_A$.

Number of spammy in-links
Top level domain of the site
Quality of phrases in the document
Density of keywords (spammy terms)

and page information. In total, we use 21 features, denoted by $FEAT_A$. Table 2 lists sample features in $FEAT_A$. The number of spammy in-links is the number of in-links coming from labeled spam pages. The quality of phrases in the document is a score that indicates the quality of terms on the page. The density of keywords is a score that indicates how many terms on the page are spam terms.

5.1 Evaluation

In spam classification, it is very important that non-spam pages not be mislabeled as spam, as typically spam detection results in demotion of a spam page in search rank, or more extreme, a removal from the search index. Quality pages must remain in the index, so we want to be sure to have high precision, and a low rate of false positives. True positives (TP) are correctly labeled spam instances. False positives (FP) are non-spam classified incorrectly as spam. True negatives (TN) are correctly labeled non-spam pages. False negatives (FN) are spam pages incorrectly classified as non-spam pages.

In our work, we use two statistics for evaluation of our classifiers: *precision*, given by $\frac{TP}{TP+FP}$, and *recall*, given by $\frac{TP}{TP+FN}$. Precision is the fraction of correctly classified spam samples out of all samples classified as spam. Recall is the fraction of correctly classified spam samples out of all spam samples. Positive labels are spam labels and negative labels are non-spam labels.

We also plot results using a receiver operating characteristic (ROC) curve. An ROC curve plots false positive rate versus recall. The false positive rate is given by $\frac{FP}{FP+TN}$. An ideal ROC curve follows the y-axis veritcally and then the x-axis horizontally, tracing the upper-left corner of the graph. A poor ROC curve follows a diagonal line (indicated by the dotted line in the figure). That is, perfect classification results in 100% recall, or sensitivity, and finds all true positives (TPs) and 100% specificity and finds no false positives (FPs).

5.2 Results

We first study the performance of a linear SVM on our set of features $FEAT_A$. This will be our baseline classifier. We determine the parameter of the linear kernel to be $C = 0.01$ by testing a range of C values during one round of cross validation. We then set C to be the same for the remaining four rounds of cross validation. We find that a linear kernel achieves a mean precision on our test sets of around 60.00% at a mean recall of 10.80%. The results are shown in Figure 3. Note that because the dataset consists of only 10% spam, the resulting precision vs. recall curve will be fairly poor. Our SVM-based classifier serves as a baseline for comparison against a classifier using additional rank-time and query-dependent features. In the remaining sections we discuss methods of improving our SVM-based classifier.

6. RANK-TIME FEATURES

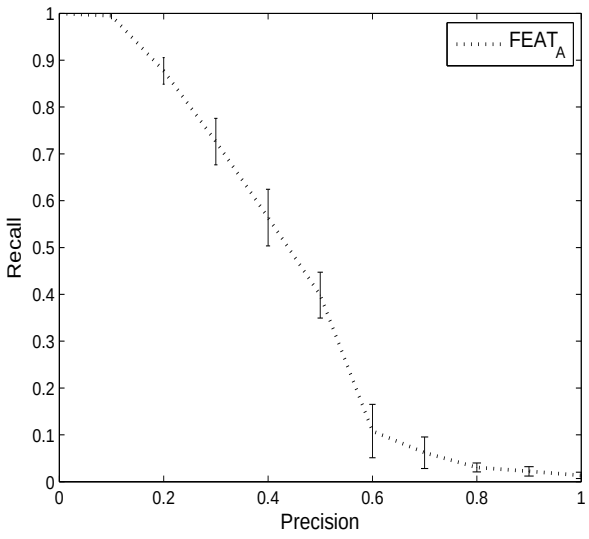

Figure 3: Precision vs. recall on our test sets using a linear SVM on our feature set $FEAT_A$.

We now examine the improvement of a spam detector which uses feature set $FEAT_A$, by adding query-independent and query-dependent rank-time features. We study the sensitivity of our results to these additional features with a linear kernel SVM. We first describe the intuition behind using rank-time features and then describe the features in detail.

6.1 Motivation

A large number of web spam pages appear in ranked search results. In order to receive a high rank, web spam pages must contain content that "fools" algorithms used for populating the index and for ranking search results. These algorithms take feature vectors as input, where the feature vectors have a specific distribution over the feature set. The distribution is difficult to match without knowledge of how the crawling, indexing, and ranking algorithms work. Even though the ranker believes the web spam page to be highly relevant, another classifier, with the same feature data as input, but trained on spam labels, should be able to easily identify web spam pages, since they will be outliers of the distribution. Since the ranker is trained to solve a different problem, namely the ordering of relevant pages, not the identification of web spam pages, by using a separate classifier trained to catch web spam, we can demote web spam in the ranked results.

Our spam classifier takes advantage of the distribution of rank-time feature values to identify web spam. Web spam pages typically display spikes in particular feature values, like keyword stuffing. These spikes will fool a linear ranking algorithm if the corresponding weight is positive, but our spam classifier can identify such characteristics because they will be outliers in our distribution of non-spam data. It is in fact most important that the web spam be outliers in the conditional distributions. That is, genuinely relevant non-spam pages will look very different from web spam pages that appear to be relevant.

Table 3: Query-independent features $FEAT_B$.

Page-level
Static rank
Most frequent term
Number unique terms
Total number of terms
Number of words in path
Number of words in title
Domain-level
Domain rank
Average number of words
Top-level domain
Popularity
Domain hits
Domain users
URL hits
URL users
Time
Date crawled
Last change date
Time since crawled

6.2 Rank-time Features

We consider 360 rank-time features. We separate the rank-time features into query-independent and query-dependent features since the query-independent rank-time features are in fact static and could be used without knowledge of the query. We denote the query-independent rank-time feature set by $FEAT_B$ and the query-dependent rank-time feature set as $FEAT_Q$.

The query-independent rank-time features can be grouped into page-level features, domain-level features, anchor features, popularity features, and time features.

Page-level features are features that can be determined by looking just at a page or URL. We use several page-level features including two static rank features, the count of the most frequent term, the count of the number of unique terms, the total number of terms, the number of words in the path, and the number of words in the title.

Domain-level features are computed as averages across all pages in a domain. Domain-level features include the rank of the domain, the average number of words, and the top-level domain.

Popularity features are features that measure the popularity of pages through user data. Our popularity features are derived from MSN Toolbar data, where the user has agreed to provide access to data collected during his logged session. Our popularity features include four domain- and page-level features. The features are the number of hits within a domain, the number of users of a domain, the number of hits on a URL, and the number of users of a URL.

Time features include the date the URL was crawled, the last date page changed, and the time since the page was crawled.

Previous studies have indicated the importance of some of the features in our feature set [11], such as frequent term counts, anchor text features, etc. Our query-independent features are listed in Table 3.

Query-dependent features are content features that relate to the query term(s). Our feature set includes 344 query-dependent features computed for each (query, URL) pair. Query-dependent features are generated from the query, document content, and URL. Query-dependent features can depend just on the query or on the match between query and document properties. Query-dependent features include the number of query terms in the title and the frequency of a query term on the page, as well as counts of the different occurrences of the query term across documents, the number of documents that contain the query term, and n-gram overlaps between the query terms and the document, for different values of n and for different skip n-grams. Table 4 lists several query-dependent features used in our classifier.

Table 4: Sample query-dependent features in $FEAT_Q$.

Number query terms in title
Freq. counts of query term in doc.
Freq. counts of query term over all docs.
Number docs. containing query term
n-grams over query terms/ doc.

Although we use query-dependent features to build our model, a spam label should not be query-dependent. Currently, however, spam is judged such that a query is issued and then each returned page is examined to determine if it is spam or not. Thus, in our datasets there are cases where a URL will have disagreeing labels for different queries. In the future, we hope to label spam based solely on the URL and not on the relevance of a URL to a query.

7. EXPERIMENTS AND RESULTS

In this section we present results for our classification model trained on the original features plus the rank-time features described in the previous section. By using rank-time features, the precision–recall curve improves. We train a SVM using a linear kernel, with $C = 0.01$, and perform five-fold cross validation. We denote the use of features in sets $FEAT_A$ and $FEAT_B$ as $FEAT_{A+B}$.

Figure 4 shows the ROC curve for a linear kernel SVM on both feature sets $FEAT_A$ and $FEAT_{A+B+Q}$. The results are micro-averaged across all five test sets. Using query-independent and query-dependent rank-time features, our ROC curve improves dramatically over the curve resulting from using the feature set $FEAT_A$.

Figure 5 shows a precision–recall curve for feature sets $FEAT_A$, $FEAT_{A+B}$, and $FEAT_{A+B+Q}$. We plot the mean recall value for precision values between 0 and 1. The error bars are determined by calculating 95% confidence intervals.

The addition of query-independent rank-time features to feature set $FEAT_A$ produces the majority of the gain in performance. We find in particular that time features and user behavior features play an important role in classification. Since web spam typically does not have a long life due to costs of domain registration and typically has fewer users, these features help discriminate between spam and non-spam pages.

The addition of query-dependent rank-time features ($FEAT_Q$) causes a further increase in performance over the feature set $FEAT_{A+B}$ in regions of high precision. At 70% precision, $FEAT_{A+B}$ yields a mean recall of 7.04% while $FEAT_{A+B+Q}$ yields a mean recall of 12.14%. Overall, by augmenting the original page-level features with rank-time features, we can

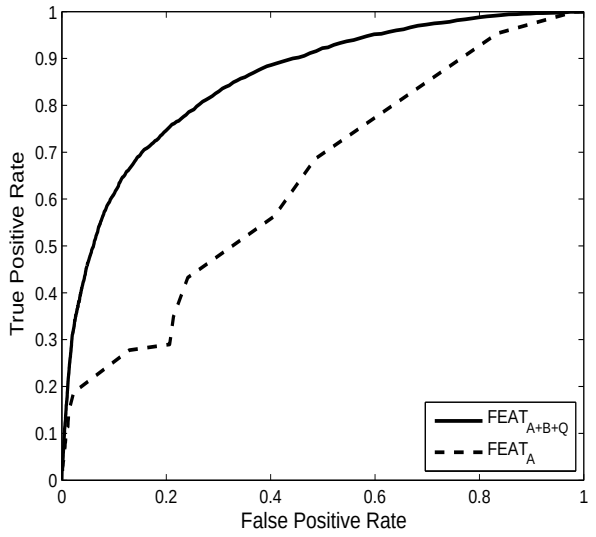

Figure 4: ROC curve on test sets using a linear SVM on features $FEAT_A$ and $FEAT_{A+B+Q}$. The dashed line represents feature set $FEAT_A$ and the solid line represents the rank-time feature set $FEAT_{A+B+Q}$.

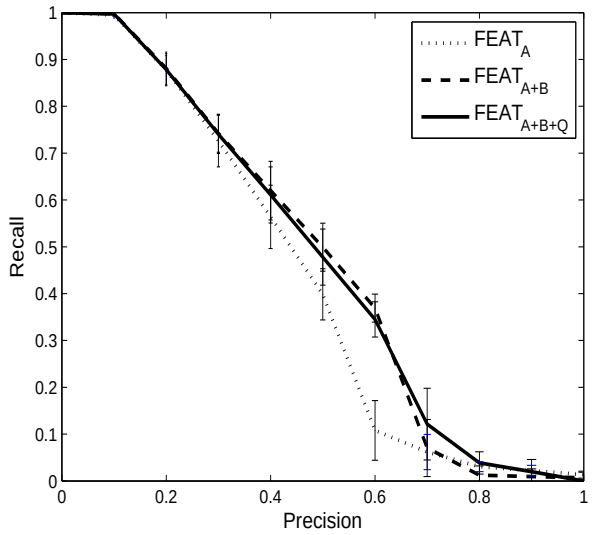

Figure 5: Precision vs. mean recall on test sets using a linear SVM on features $FEAT_A$, $FEAT_{A+B}$, and $FEAT_{A+B+Q}$.

Table 5: Mean recalls at 60% precision.

Feature Set	Precision	Recall	Std. Error
$FEAT_A$	60.0%	10.8%	5.98%
$FEAT_{A+B}$	60.0%	36.91%	2.82%
$FEAT_{A+B+Q}$	60.0%	34.49%	3.53%

improve the recall by around 25% for a similar precision. Our results may be further improved by using non-query-based spam judgements.

8. RELATED WORK

Our work is related to many previous studies of feature-based web spam classification. Since the beginning of the World Wide Web, there has been a need to rank pages according to relevance to a given query. However the generation of large amounts of revenue from rank-related advertising, and the reliance of businesses on the ranking of their pages for exposure to customers, is only a relatively recent development. These significant opportunities for new business models come at the cost of a high reward, and low bar (in terms of cost), for unscrupulous players hoping to tilt the game in their favor, at the expense of the common good. With such high incentives, black and grey hat SEOs have rapidly learned to use increasingly sophisticated techniques. The response from the academic community has ramped up only recently: in [9], Henzinger *et al.* declared web spam classification to be one of the most important challenges to search engines. In particular, they discuss the degradation of search engine quality due to web spam. Their paper sparked a significant increase in interest of web spam classification in the academic community.

One promising method of identifying web spam uses link information. Link spam stuffs pages with extraneous links to web pages or adds extraneous pages with links to other pages to increase the size of the *link farm* and the rank of pages. Davison investigates link spam in [5]. Amitay *et al.* used feature-based methods to identify link spam [1]. Many other papers have investigated link spam and developed classification methods to detect link farms [2, 7, 15].

In addition, content-based web spam classification has been the recent subject of many studies. In [11], Ntoulas *et al.* studied specific content-based features. In [6], Fetterly *et al.* examine statistical properties of web pages to devise a set of content-based features for use in a web spam classifier. Blog spam detection is studied in [10].

More recently, there has been much work on detecting large amounts of web spam by discovering doorway pages, or cloaking methods. Cloaking is when one version of a page is shown to the user and another version of the page is shown to the crawler. Recently, in [14], cloaking methods have been detected through examining three copies of a page. In [13], Wang *et al.* propose to identify redirection spam pages by connecting spammers and advertisers through redirection analysis.

All of these responses are just first steps in combating web spam: the necessarily adversarial nature of the task results in a rapidly evolving problem, and this property (of finding techniques that succeed in the face of adversarial adaptation) is new to many in the machine learning community and brings with it many new challenges.

9. CONCLUSIONS

In this paper we have studied the sensitivity of a web spam classifier to rank-time features. Through numerous studies based on real-world datasets, we have demonstrated the necessity of evaluating web spam classifiers on domain-separated data. We note that the problem of intermediate structures in the data, which can mislead workers into believing overly optimistic results, is likely to exist in the more general machine learning setting as well and we note that this may be a fruitful area for research. Our results indicate that an extended set of features based on rank-time content is more effective in classifying web spam pages than a purely page-based feature set. Our experiments confirm our intuition that the goal of spammers to generate data to fool search engine ranking algorithms can lead to tell-tale properties of their data that can be detected by secondary classifiers.

One advantage that search engine companies have, that the spammers don't, is large quantities of data: it will be difficult for spammers to construct pages (and associated link structure) that will not become outliers in the distribution of features. As classifiers earlier in the chain of spam classifiers improve, and as more spam pages are removed from the index, we expect the precision rates of our classifier to degrade, since only "hard" web spam will remain. We view this as a positive development and will face the challenge of determining even better differentiating features in the future.

The increasing importance of search for business needs, and the increasing amounts of revenue generated by online advertising, lead us to believe that the number of spam attacks is likely to go on increasing for the foreseeable future. As search engine designers develop new web spam classifiers, spammers will also derive new techniques for "beating" the ranking algorithms. There will continue to be an "arms race" between spammers and search engine designers. The focus of our work is to improve a user's search engine experience by removing as much web spam as possible from search results and to increase the potential profits of legitimate pages. Ultimately, a search engine employs web spam classifiers to enhance and protect the online experience of legitimate users and businesses.

10. ACKNOWLEDGMENTS

We would like to thank Andrey Zaytsev, Jacob Richman, and Matt Richardson for assisting in obtaining our datasets. We would also like to thank Andy Laucius for many fruitful discussions.

11. REFERENCES

[1] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: Detecting site functionality by structural patterns. In *14th ACM Conference on Hypertext and Hypermedia*, 2003.

[2] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *Proceedings of the Workshop on Web Mining and Web Usage Analysis (WebKDD)*. ACM Press, August 2006.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. *Learning to Rank using Gradient Descent*. Bonn, Germany, 2005.

[4] C. Castillo, D. Donato, L. Becchetti, P. Boldi, M. Santini, and S. Vigna. A reference collection for web spam. In *SIGIR Forum*, volume 40, December 2006.

[5] B. Davison. Recognizing nepotistic links on the web. In *Artificial Intelligence for Web Search*, pages 23–28. AAAI Press, 2000.

[6] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In S. Amer-Yahia and L. Gravano, editors, *WebDB*, pages 1–6, 2004.

[7] Z. Gyongyi and H. Garcia-Molina. Link spam alliances. In *Proceedings of the 31st VLDB Conference*, 2005.

[8] Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy. In *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, 2005.

[9] M. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. In *Proc. of the 18th International Joint Conference on Artificial Intelligence*, pages 1573–1579, 2003.

[10] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, 2005.

[11] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 83–92. ACM, 2006.

[12] V. Vapnik. *The Nature of Statistical Learning*. Springer-Verlag, 1995.

[13] Y. Wang, M. Ma, Y. Niu, and H. Chen. Spam double-funnel: Connecting web spammers with advertisers. In *Proc. of International World Wide Web (WWW)*, May 2007.

[14] B. Wu and B. Davison. Cloaking and redirection: a preliminary study. In *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, May 2005.

[15] B. Wu and B. Davison. Identifying link farm spam pages. In *Proceedings of the 14th International World Wide Web Conference, Industrial Track*, May 2005.