J Comput Virol
DOI 10.1007/s11416-009-0132-6

ORIGINAL PAPER

Removing web spam links from search engine results

Manuel Egele · Clemens Kolbitsch · Christian Platzer

Received: 22 December 2008 / Accepted: 3 August 2009

Abstract Web spam denotes the manipulation of web pages with the sole intent to raise their position in search engine rankings. Since a better position in the rankings directly and positively affects the number of visits to a site, attackers use different techniques to boost their pages to higher ranks. In the best case, web spam pages are a nuisance that provide undeserved advertisement revenues to the page owners. In the worst case, these pages pose a threat to Internet users by hosting malicious content and launching drive-by attacks against unsuspecting victims. When successful, these drive-by attacks then install malware on the victims' machines. In this paper, we introduce an approach to detect web spam pages in the list of results that are returned by a search engine. In a first step, we determine the importance of different page features to the ranking in search engine results. Based on this information, we develop a classification technique that uses important features to successfully distinguish spam sites from legitimate entries. By removing spam sites from the results, more slots are available to links that point to pages with useful content. Additionally, and more importantly, the threat posed by malicious web sites can be mitigated, reducing the risk for users to get infected by malicious code that spreads via drive-by attacks.

M. Egele (✉) · C. Kolbitsch · C. Platzer
Vienna University of Technology, Vienna, Austria
e-mail: manuel@seclab.tuwien.ac.at

C. Kolbitsch
e-mail: ck@seclab.tuwien.ac.at

C. Platzer
e-mail: cplatzer@seclab.tuwien.ac.at

Published online: 22 August 2009

1 Introduction

Search engines are designed to help users find relevant information on the Internet. Typically, a user submits a query (i.e., a set of keywords) to a search engine, which then returns a list of links to pages that are most relevant to this query. To determine the most-relevant pages, a search engine selects a set of candidate pages that contain some or all of the query terms and calculates a *page score* for each page. Finally, a list of pages, sorted by their score, is returned to the user.

This score is calculated from properties of the candidate pages, so-called features. Unfortunately, details on the exact algorithms that calculate these ranking values are kept secret by search engine companies, since this information directly influences the quality of the search results. Only general information is made available. For example, in 2007, Google claimed to take more than 200 features into account for the ranking value [9].

The way in which pages are ranked directly influences the set of pages that are visited frequently by the search engine users. The higher a page is ranked, the more likely it is to be visited [4]. This makes search engines an attractive target for everybody who aims to attract a large number of visitors to her site. There are three categories of web sites that benefit directly from high rankings in search engine results. First, sites that sell products or services. In their context, more visitors imply more potential customers. The second category contains sites that are financed through advertisement. These sites aim to rank high for any query. The reason is that they can display their advertisements to each visitor, and, in turn, charge the advertiser. The third, and most dangerous, category of sites that aim to attract many visitors by ranking high in search results are sites that distribute malicious software. Such sites typically contain code that exploits web browser vulnerabilities to silently install

malicious software on the visitor's computer. Once infected, the attacker can steal sensitive information (such as passwords, financial information, or web-banking credentials), misuse the user's bandwidth to join a denial of service attack, or send spam. The threat of drive-by downloads (i.e., automatically downloading and installing software without the user's consent as the result of a mere visit to a web page) and distribution of malicious software via web sites has become a significant security problem. Web sites that host drive-by downloads are either created solely for the purpose of distributing malicious software or existing pages that are hacked and modified (for example, by inserting an `iframe` tag into the page that loads malicious content). Provos et al. [20,21] observe that such attacks can quickly reach a large number of potential victims, as at least 1.3% of all search queries directed to the Google search engine contain results that link to malicious pages. Moreover, the pull-based infection scheme circumvents barriers (such as web proxies or NAT devices) that protect from push-based malware infection schemes (such as traditional, exploit-based worms). As a result, the manipulation of search engine results is an attractive technique for attackers that aim to attract victims to their malicious sites and spread malware via drive-by attacks [23].

Search engine optimization (SEO) companies offer their expertise to help clients improve the rank for a given site through a mixture of techniques, which can be classified as being acceptable or malicious. Acceptable techniques refer to approaches that improve the content or the presentation of a page to the benefit of users. Malicious techniques, on the other hand, do not benefit the user but aim to mislead the search engine's ranking algorithm. The fact that bad sites can be pushed into undeserved, higher ranks via malicious SEO techniques leads to the problem of *web spam*.

Gyöngyi and Garcia-Molina [10] define web spam as every deliberate human action that is meant to improve a site's ranking without changing the site's true value. Search engines need to adapt their ranking algorithms continuously to mitigate the effect of spamming techniques on their results. For example, when the Google search engine was launched, it strongly relied on the PageRank [3] algorithm to determine the ranking of a page where the rank is proportional to the number of incoming links. Unfortunately, this led to the problem of link farms and "Google Bombs," where enormous numbers of automatically created forum posts and blog comments were used to promote an attacker's target page by linking to it.

Clearly, web spam is undesirable, because it degrades the quality of search results and draws users to malicious sites. Although search engines invest a significant amount of money and effort into fighting this problem, checking the results of search engines for popular search terms demonstrates that the problem still exists. In this work, we aim to post-process results returned by a search engine to identify entries that link to spam pages. To this end, we first study the importance of different features for the ranking of a page. In some sense, we attempt to reverse-engineer the "secret" ranking algorithm of a search engine to identify the most important features. Based on this analysis, we attempt to build a classifier that inspects these features to identify indications that a page is web spam. When such a page is identified, we can remove it from the search results.
The two main contributions of this paper are the following:

- We conducted comprehensive experiments to understand the effects of different features on search engine rankings.
- We developed a system that allows us to reduce spam entries from search engine results by post-processing them. This protects users from visiting either spam pages or, more importantly, malicious sites that attempt to distribute malware.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of our overall approach. In Sect. 3, we discuss our experiment that helped us understand how the ranking is generated by major search engines. Then, we describe our system for detecting web spam in search engine results and examines its effectiveness. Section 5 presents related work, and Sect. 6 briefly concludes.

2 Overview

In this section, we first provide an overview of our approach to determine the features that are important for the ranking algorithm. Then, we describe how we use this information to develop a technique that allows us to identify web spam pages in search engine results.

2.1 Inferring important features

Unfortunately, search engine companies keep their ranking algorithms and the features that are used to determine the relevance of a page secret. However, to be able to understand which features might be abused by spammers and malware authors to push their pages, a more detailed understanding of the page ranking techniques is necessary. Thus, the goal of the first step of our work is to determine the features of a web page that have the most-pronounced influence on the ranking of this page.

A feature is a property of a web page, such as the number of links pointing to other pages, the number of words in the text, or the presence of keywords in the title tag. To infer the importance of the individual features, we perform "black-box testing" of search engines. More precisely, we create a set of different test pages with different combinations of features and observe their rankings. This allows us to

deduce which features have a positive effect on the ranking and which contribute only a little.

2.2 Removing spam from search engine results

Based on the results of the previous step, we developed a system that aims to remove spam entries from search engine results. To this end, we examine the results that are returned by a search engine and attempt to detect links that point to web spam pages. This is a classification problem; every page in the result set needs to be classified as either spam or nospam.

To perform this classification, we have to determine those features that are indicators of spam. For this, we leverage the findings from the first step and a labeled training set to construct a C4.5 decision tree. A decision tree is useful because of its intuitive insight into which features are important to the classification. Using this classifier, we can then check the results from the search engine and remove those links that point to spam pages. The result is an improvement of search quality and fewer visits to malicious pages.

3 Feature inference

In this section, we give a detailed introduction to our inference techniques for important features. First, we discuss which features we selected. Then, we describe how these features are used to prepare a set of (related, but different) pages. Finally, we report on the rankings that major search engines produced for these pages and the conclusions that we could draw about the importance of each feature.

3.1 Feature selection

As mentioned previously, we first aim to "reverse engineer" the ranking algorithm of a search engine to determine those features that are relevant for ranking.

Based on reports from different SEO vendors [24] and study of related work [2,6], we chose ten presumably important page features (see Table 1). We focused on features that can be directly influenced by us. The rationale is that only from the exact knowledge of the values of each feature, one can determine their importance. Additionally, the feature value should remain unchanged during the whole experiment. This can only be ensured for features under direct control.

When considering features, we first examined different locations on the page where a search term can be stored. Content-based features, such as body-, title-, or headings-tags are considered since these typically provide a good indicator for the information that can be found on that page. Additionally, we also take link-based features into account (since search engines are known to rely on linking information). Usually, the number of incoming links pointing to a page (i.e., the *in-link* feature) cannot be influenced directly. However, by recruiting 19 volunteers willing to host pages linking to our experiments, we were able to fully control this feature as well.

Table 1 Feature set used for inferring important features

1	Keyword(s) in title tag
2	Keyword(s) in body section
3	Keyword(s) in H1 tag
4	External links to high quality sites
5	External links to low quality sites
6	Number of inbound links
7	Anchor text of inbound links contains keyword(s)
8	Amount of indexable text
9	Keyword(s) in URL file path
10	Keyword(s) in URL domain name

Together with features that are not directly related to the page's content (e.g., keyword in domain name), we believe to have covered a wide selection of features from which search engines can draw information to calculate the rankings.

We are aware of the fact that search engines also take temporal an location-based aspects into account when computing their rankings (e.g., how does a page or its link count evolve over time). However, we decided against adding time- and location-dependent features to our feature set because this would have made the experiment significantly more complex and does not necessarily add value. We aim to eliminate the influence of these aspects by always publishing and modifying pages from the same location and at the same time.

3.2 Preparation of pages

Once the features were selected, the next step was to create a large set of test pages, each with a different combination and different values of these features. For these test pages, we had to select a combination of search terms (a query) for which no search engine would produce any search results prior to our experiment (i.e., only pages that are part of our experiment are part of the results). We arbitrarily chose "gerridae plasmatron" as the key phrase to optimize the pages for.[1] Remember, the goal is to estimate the influence of page features to the ranking algorithms and not to determine whether our experiment pages outperform (in terms of search engine response position) existing legitimate sites.

Using this search phrase, we prepared the test pages for our experiment. To this end, we first created a reference page

[1] Gerridae is the Latin expression for water strider, plasmatron is a special form of an ion source.

consisting of information about gerridae and plasmatrons compiled from different sources. In a second step, this reference page was copied 90 times. To evade duplicate detection by search engines (where duplicate pages are omitted from the results), each of these 90 pages was obfuscated by substituting many words in a manner similar to [15]. Subsequent duplicate detection by the search engines (presumably based on title and headline tag) required a more aggressive obfuscation scheme where title texts and headlines where randomized as well.

For features whose possible values exceed the boolean values (i.e., present or absent), such as keyword frequencies, we selected representative values that correspond to one of the following four classes.

- The feature is not present at all.
- The feature is present in *normal* quantities.
- The feature is present in *elevated* quantities.
- The feature is present in *spam* quantities.

That is, a feature with a large domain (i.e., set of possible values) can assume four different values in our experiment. Of course, there is no general rule to define a precise frequency for which a feature can be considered to be normal, elevated, or spam. Thus, we manually examined legitimate and spam pages and extracted average, empirical frequencies for the different values. For example, for the frequencies of the keyword in the body text, a 1% keyword frequency is used as a baseline, 4% is regarded elevated, and 10% is considered to be spam.

Since only 90 domains were available, we had to select a representative subset of the 16,392 possible feature combinations. Moreover, to mitigate any measurement inaccuracies, we decided to do all experiments triple-redundant. That is, we chose a subset of 30 feature combinations, where each combination forms an experiment group that consists of three identical instances that share the same feature values. For these 30 experiment groups, we decided to select the feature values in a way to represent different, common cases. The regular case is a legitimate site, which is represented by the reference page. For this page, all feature values belong to the *normal* class. Other cases include keyword stuffing in different page locations (e.g., body, title, headlines), or differing amounts of incoming and outgoing links. The full list of the created experiments can be found in Appendix B.

3.3 Execution of experiments and results

Once the 30 experiment groups (i.e., 90 pages) were created, they were deployed to 90 freshly registered domains, served by four different hosting providers. Additionally, some domains were hosted on our department web server. This was done to prevent any previous reputation of a long-lived domain to influence the rankings, and hence, our results.

Once the sites were deployed, we began to take hourly snapshots of the search engine results for the query "gerridae plasmatron." To keep the results comparable we queried the search engines for results of the English web (i.e., turning off any language detection support). In addition, we also took snapshots of results to queries consisting of the individual terms of the key phrase. Since all major search engines had results for the single query terms (gerridae/plasmatron) before our experiment started, we gained valuable insights into how our sites perform in comparison to already existing, mostly legitimate sites.

Our experiment was carried out between December 2007 and March 2008. During 86 days, we submitted 2,312 queries to Google and 1,700 queries to the Yahoo! search engine. Interestingly, we observed that rankings usually do not remain stable over a longer period of time. In fact, the longest period of a stable ranking for all test pages was only 68 h for Google and 143 h for Yahoo!. Also, we observed that Google refuses to index pages whose path (in the URL) contained more than five directories. This excluded some of our test pages from being indexed for the first couple of weeks.

One would expect that instances within the same experiment group occupy very close positions in the search engine results. Unfortunately, this is not always the case. While there were identical instances that ranked at successive or close positions, there were also some experiment groups whose instances were significantly apart. We suspect that most of these cases are due to duplicate detection (where search engines still recognized too many similarities among these instances).

At the time of writing, querying Google for "gerridae plasmatron" resulted in 92 hits. Including omitted results, 330 hits are returned. Yahoo! returns 82 hits without and 297 hits including the omitted results. Microsoft Live search returns only 28 pages. Since Microsoft Live search seemed slower in indexing our test pages, we report our results only for Google and Yahoo!.

Note that the Google and Yahoo! results consist of more than 90 elements. The reason for this is that the result sets also contain some sites of the volunteers, which frequently contain the query terms in anchor texts pointing to the test sites.

For Google, searching for "gerridae" yields approximately 55,000 results. Our test pages constantly managed to occupy five of the top ten slots with the highest ranking page at position three. Six was the highest position observed for the "plasmatron" query.

For Yahoo!, we observed that for both keywords pages of our experiments managed to rank at position one and stay there for about two weeks.

3.4 Extraction of important features

Because of the varying rankings, we determined a page's position by averaging its positions over the last six weeks of the experiment. We decided for the last six weeks, since the initial phase of our experiment contains the inaccuracies that were introduced due to duplicate detection. Also, it took some time before most pages were included in the index. We observed that when we issued the same query to Google and Yahoo!, they produced different rankings. This indicates that the employed algorithms weight features apparently differently. Thus, we extracted different feature weights for Google and Yahoo! as described below.

Knowing the combinations of all feature values for a page k and observing its position $pos(k)$ in the rankings, our goal is now to assign an (optimal) weight to each feature that best captures this feature's importance to the ranking algorithm. As a first step, we define a function $score$. This function takes as input a set of weights and feature values and computes a score $score(k)$ for a page k.

$$score(k) = \sum_{i=1}^{n} f_i^k \cdot w_i$$

$$\begin{aligned} n, \ldots, &\text{ number of features} \\ w_i \in [-1, 1], \ldots, &\text{ weight of feature } i \\ f_i^k \in 0, 1, \ldots, &\text{ presence of feature } i \text{ in test page } k \end{aligned}$$

This calculation is repeated for all test pages (of course, using the same weights). Once all scores are calculated, the set of test pages is sorted by their score. This allows us to assign a predicted ranking $rank(k)$ to each page. Subsequently, distances between the predicted ranking and the real position are calculated for all test pages. When the sum of these distances reaches the minimum, the weights are optimal. This translates to the following objective function of a linear programming problem (LP):

$$\min : \sum_{k=1}^{m} \alpha_k |pos(k) - rank(k)|$$

Note that we added the factor $\alpha(k) = m - pos(k)$ to the LP, which allows higher-ranking test pages to exert a larger influence on the feature weights (m is the number of test pages). This is to reflect that the exact positions of lower-ranking pages often fluctuated significantly. Thus, we required a way to reduce these "random" influence on the calculation of the weights. Solving this LP with the Simplex algorithm results in weights for all features that, over all pages, minimize the distance between the predicted rank and the actual position.

For Google, we found that the number of search terms in the title and the text body of the document had the strongest, positive influence on the ranking. Also, the number of outgoing links was important. On the other hand, the fact that the keywords are part of the file path had only a small influence. This is also true for the anchor text of inbound links.

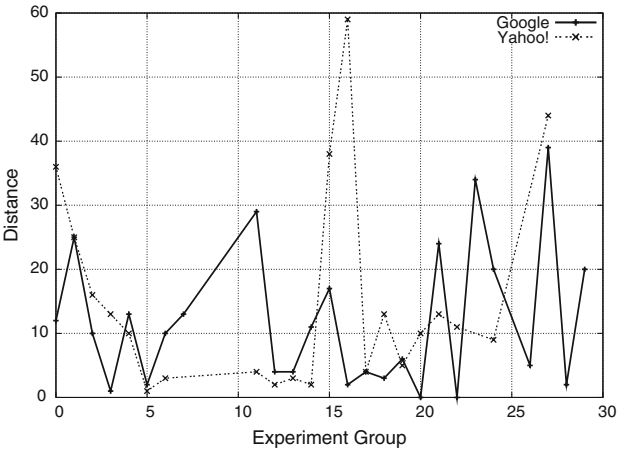

Fig. 1 Differences when comparing predicted values with actual ranking positions

For Yahoo!, the features were quite different. For example, the fact that a keyword appears in the title has less influence and even decreases with an increase of the frequency. Yahoo! also (and somewhat surprisingly) puts significantly more weight on both the number of incoming and outgoing links than Google. On the other hand, the number of times keywords appear in the text have no noticeable, positive effect.

As a last step, we examine the quality of our predicted rankings. To this end, we calculate the distance between the predicted position and the actual position for each experiment group. More precisely, Fig. 1 shows, for each experiment group, the distance between the actual and predicted positions, taking the closest match for all three pages in each group.

Considering the Google results, 78 experiment pages of 26 experiment groups were listed in the rankings. The missing experiment groups are those whose pages have a directory hierarchy level of five, and thus, were not indexed by the search engine spiders. Looking at the distance, we observe that we can predict the position for six groups (23%) within a distance of two, and for eleven groups (42%) with a distance of five or less (over a range of 78 positions). For Yahoo!, when comparing the experiment groups with the rankings, 21 groups appear in the results. Three (14%) of these groups are predicted within a distance of two, while eight (38%) are within a distance of five or less positions to the observed rank (over a range of 63 positions).

At a first glance, our predictions do not appear very precise. However, especially for Yahoo!, almost all predictions are reasonably close to the actual results. Also, even though our predictions are not perfectly accurate, they typically reflect the general trend. Thus, we can conclude that our

general assessment of the importance of a feature is correct, although the precise weight value might be different. Also, we only consider a linear ranking function, while the actual ranking algorithms are likely more sophisticated.

4 Reducing spam from search engine results

In this section, we present the details of our prototype system to detect web spam entries in search engine results. The general idea behind this system is to use machine learning techniques to generate a classification model (a classifier) that is able to distinguish between legitimate and spam sites by examining a page's features. The following section first presents the details on how the system operates. Then, the evaluation section describes our spam detection effectiveness.

4.1 Detecting web spam in search engine results

During the previous feature inference step, we determined the features that are most important to search engine ranking algorithms. Assuming an attacker can also learn this information, this suggests that the attacker will focus on those features that have the most pronounced influence on the rankings. This motivates our approach in developing a classifier that distinguishes spam and non-spam pages according to these features.

The classifier presented in this section is developed for the Google search engine. Thus, we include those features that are most relevant for Google, as discussed in the previous section. These are the number of keywords in the title, body, and domain name. In addition, we consider linking information. While counting the outgoing links of a page is trivial, the number of incoming links is not easily determinable. The information of how many in-links point to a page is not made available by search engines. This is the reason why we have to estimate the corresponding features with the help of `link:` queries. Google and Yahoo! support queries in the form of `link:`http://www.example.com resulting in a list of pages that link to http://www.example.com. The drawback is that neither the Google nor the Yahoo! results contain all pages that link to the queried page. Thus, these numbers are only an approximation of the real number of links pointing to a site.

On the other hand, we can introduce additional information sources that were not available to us before. For example, the PageRank value (as reported by the Google toolbar) was added to the feature set. This value could not be used for the experiment because of the infrequent updates (roughly every three months) and its violation of the requirement that we can control each feature directly.

4.1.1 Classifier

To build a classifier for web pages, we first require a labeled training set. Another set of data is required to verify the resulting model and evaluate its performance. To create these sets, 12 queries were submitted to the Google search engine (asking for popular search terms, extracted from Google's list of popular queries, called Zeitgeist [8]). For every query, the first 50 results were manually classified as legitimate or spam/malicious. Discarding links to non-HTML content (e.g., PDF or PPT files) resulted in a training data set consisting of 295 sites (194 legitimate, 101 spam). The test data set had 252 pages (193 legitimate, 59 spam).

All result pages were downloaded and fed into feature extractors that parse the HTML source code and return the value (i.e., the frequency) of the feature under consideration. If the query consists of multiple terms, query dependent feature extractors report higher values if the full query matches the analyzed feature. The rationale behind this is that a single heading tag that contains the whole query indicates a better match than multiple, individual heading tags, each containing one of the query terms. Feature extractors that follow this approach are marked with an (X) in the following list, which enumerates all the features that we consider:

- **Title:** the number of query terms from HTML `title` tag (X)
- **H1-Tag:** the number of query terms in HTML `H1`-tags (X)
- **Body:** the number of query terms in the HTML `body` section (X)
- **Domain name:** the number of query terms in the domain name part of the URL (e.g, http://www.gerridae-plasmatron.com/index.php)
- **Filepath:** the number of query terms in the path of the URL (e.g., http://www.example.org/gerridae-plasmatron/index.php)
- **Out-links:** the total number of outbound links
- **Out-links-keywords:** number of outbound links containing keywords as anchor texts (X)
- **In-links-Google:** the number of inbound links reported by Google `link:` query
- **In-links-Yahoo!:** the number of inbound links reported by Yahoo! `link:` query
- **PageRank site:** the Google PageRank value for the URL as reported by the Google toolbar
- **PageRank domain:** the Google PageRank value for the domain as reported by the Google toolbar
- **Word count:** total number of words in the document
- **Tfreq:** the frequency of query terms appearing on the page (number of query terms/number of words on page)

Using the labeled training data as a basis, we ran a series of algorithms to train different classification models. Each classifier was evaluated against the test data set. To this end, we leveraged the Weka [28] toolkit that provides support for a multitude of classification models.

4.2 Evaluated classification models

We evaluated a total of eight distinct classification models from the *Weka* toolkit to assess their applicability for our purpose. Each model comes with a unique set of properties and settings, which we briefly discuss here. A comparative evaluation of the presented classification models is illustrated in Table 2.

4.2.1 Naive Bayesian classifier

Heckerman [12] describes Bayesian networks as a graphical model that encodes probabilistic relationships among variables of interest. Such a network can be learned from training data or a-priori knowledge. In our case, the model is inferred from our training data set. The applied naive Bayesian classifier is described by John et al. [13] as a Bayesian network that relies on two simplifying assumptions. First, the predictive attributes for a given class are conditionally independent. Furthermore, no hidden or latent attributes influence the prediction process.

4.2.2 Support vector machines with sequential minimal optimization

Support vector machines [11] (SVMs) can classify objects by projecting them into a n-dimensional space. The dimensional size is determined by the number of characteristics of the training- or query-vector. In our case, the feature vector is of size 13, resulting in an equally sized vector space. The actual classification is done by filling the vector space with labeled elements from the training set and creating a hyperplane that separates the points according to their labels. A query can then be categorized by simply projecting it into the same space and determining on which side of the plane it resides.

Platt [19] introduced sequential minimal optimization (SMO) as an alternative approach to train support vector machines. The original SVM's training step requires that a large quadratic programming problem is solved. SMO speeds up this training by breaking the QP problem into a series of smallest possible QP problems which can be solved analytically. Once the SVM is trained, classification is performed like in a normal SVM.

4.2.3 Locally weighted learning

As explained in [1], the locally weighted learning approach can be used to classify an unknown sample by the following steps. Given the sample point to classify, the system calculates those points in the training data set that are close to this sample. The class of each of those points influences the classification decision inversely proportional to the measured distance. That is, the closer a point in the training set is to the sample to classify, the more influence this point's class has on the final classification decision.

4.2.4 Fuzzy lattice reasoning classifier (FLR)

During training, this classification model [14] generates hyperboxes based on the points contained in the training data set. A hyperbox corresponds to a rule which indicates that a point located within this box is a member of the respective class. Training is performed iteratively and each point from the training data set induces a new rule. For each new rule the model calculates a fuzzy degree of inclusion with the existing rules. The maximum value of these degrees suggests, how existing and new rules must be combined. For classifying an unknown data point, the system calculates the inclusion degrees for the rule induced by the unknown point and assigns this point to the class of the rule with the highest value.

4.2.5 ConjunctiveRule

This classifier aims to make decisions based on a single logic proposition of the form `if` A`,` `then` B. The proposition's `antecedent` (that is, the A part of the proposition) is the conjunction of selected feature rules, whereas the `consequent` is the mere statement whether a site is to be labeled as spam or not.

The proposition is generated by iteratively selecting single features from the data set. The algorithm greedily adds feature rules based on their *information gain* calculated on the instances of the training set that are not covered by the current rule. To avoid over-fitting of possibly irrelevant features, the Weka toolkit uses *reduced error pruning* in its default configuration.

An exemplary logic proposition, generated from our training set, can be seen below:

$$(title < 0.166667) \wedge (domainname \leq 1.5) \wedge (tfreq \leq 3.5) \rightarrow \textit{no-spam}$$

4.2.6 J48

The `J48` classifier implements the $C4.5$ algorithm [22] for generating decision trees. Given a set of features, the

Table 2 Comparison of different classification models

Model	TP	FN	FP	TN	FPR	Precision	Recall
bayes.NaiveBayes	22	37	29	164	0.15	0.43	0.37
function.SMO	19	40	27	166	0.13	0.41	0.32
lazy.LWL	28	31	33	160	0.17	0.45	0.47
misc.FLR	2	57	9	184	0.04	0.18	0.03
rules.ConjunctiveRule	37	22	68	125	0.35	0.35	0.62
trees.J48	28	31	46	147	0.23	0.37	0.47
trees.BFTree	24	35	35	158	0.18	0.40	0.40
meta.Clustering	30	29	61	132	0.31	0.32	0.50

heuristic selects an item that best splits the training set into distinct groups. To this end, the *normalized information gain* is leveraged to compare all available features. The selected feature (i.e., the feature with highest gain) is inserted as a rule into a decision tree, dividing the data set into distinct subsets.

This classification is repeated on the generated subsets until all features have been mapped as rules into the decision tree or none of the remaining features can be used to correctly sub-classify the data. Terminal nodes are inserted into the tree holding the information on branch classification (such as *spam* or *no-spam*).

A more detailed description of this method can be found in Sect. 4.3. Furthermore, Appendix A shows a $C4.5$ decision tree generated from the training set.

4.2.7 Best-first decision tree

This decision tree heuristics differs from the `J48` classifier by means of selecting the *ordering* of features used for making a decision. The *best-first* heuristics [25] seeks to maximally reduce impurity of subsets after each decision.

That is, the features are selected in such a way that the branch-paths are as short as possible, ending in *pure* terminal nodes (all elements of that node have been equally classified) as soon as possible.

4.2.8 Clustering

Weka offers a multitude of algorithms for computing clusters. The most widely used method is the *k-means clustering* [16]. Here, the training set is organized in d-dimensional vectors. k randomly selected vectors make up the mean values for *initial* clustering. The remaining vectors are assigned to the closest cluster based on the *euclidean distance* to the cluster's selected mean value.

These initial clusters are refined by updating the chosen means with their actual values, possibly reassigning elements to clusters with closer distance. This refinement is repeated until no further modifications in the partitioning/clustering of the vectors is observed.

Thus, the training set is always divided into 2 clusters ($k = 2$) containing 13-dimensional vectors ($d = 13$), one dimension per feature.

Table 2 gives a numerical representation of the classification approaches presented in this section. The first four columns entitle the number of *true/false positives* (TP/FP) and *true/false negatives* (TN/FN), followed by the *false positive rate*. The numbers show, that most of the approaches perform comparably good in terms of precision and recall, with Fuzzy Lattice Reasoning (misc.FLR) as the only exception.

4.3 Evaluation of the J48 decision tree

We chose a decision tree as the classifier because it intuitively presents the importance of the involved features (i.e., the closer to the root a feature appears in the tree, the more important it is). The J48 implementation included in the Weka package, offers various possibilities to tweak the final result. The most interesting parameter is the confidence factor, which allows to tweak the degree of pruning and, therefore, classification accuracy. We found that a value of 0.1 leads to the best results for our data set. The generated tree is shown in Appendix A. This tree consists of 21 nodes, 11 of which are leafs. Five features were selected by the algorithm to be useful as distinction criteria between spam and legitimate sites. Additionally, Weka calculates a confidence factor for every leaf, indicating how accurate this classification is.

The most important feature is related to the presence of the search terms on the page (i.e., the query term frequency >0). Other important features are the domain name, the file path, the number of in-links as reported by Yahoo!, and the PageRank value of the given site as reported by the Google toolbar.

The fact that we want to improve the results by removing spam sites demands a low false positive rate. False positives are legitimate sites that are removed from the results because they are misclassified as spam. It is clearly desirable to have a

Table 3 Confusion matrix of the J48 decision tree

	Classified as spam	Classified as legitimate
Spam	21	38
Legitimate	20	173

low number of these misclassifications, since false positives influence the quality of the search results in a negative way. False negatives on the other hand, do not have an immediate negative effect on the search results. If a spam site is misclassified as legitimate, it ends up as part of the search results. Since we are only post-processing search engine results, the site was there in the first place. Thus, false negatives indicate inaccuracies in our classification model, but do not influence the quality of the original search results.

Evaluating the J48 decision tree with our test data set results in the confusion matrix as shown in Table 3. The classifier has a false positive rate of 10.8% and a false negative rate of 64.4%. The detection rate (true positives) is 35.6%.

Detecting 35% of the unwanted sites is good, but the false positive rate of 11% is unacceptable. To lower the false positive rate, we decided to incorporate the confidence factor that is provided for each leaf in the decision tree. By using this confidence factor as a threshold (i.e., a site is only classified as spam when the confidence factor is above the chosen threshold), we can tune the system in a way that it produces less false positives, at the cost of more false negatives. For example, by using a confidence value of 0.88, the classifier has a false negative rate of 81.4%. However, it produces no false positives for our test set. The true positive rate with this threshold value is 18.6%, indicating that the system still detects about every fifth spam/malicious page in the search results.

While a detection rate of 18% is not perfect and allows for improvement, it clearly lowers the amount of unwanted pages in the results. Taking into consideration that most users only pay attention to the top 10 or top 20 results of a search query, these 18% create up to two empty slots in the top 10 rankings that can accommodate potentially interesting pages instead.

5 Related work

In recent years, considerable effort was dedicated to the detection and mitigation of web spam. In [10], the authors present different techniques to fool search engine ranking algorithms. Boosting techniques, such as link farms, are used to push pages to undeserved higher ranks in search engine results. Hiding or cloaking techniques are used to trick search engines by serving different content to the search engine spiders and human users.

One of the most prominent boosting techniques are link farms, and multiple researchers have presented techniques for detecting them. For example, Wu and Davison [30] propose an algorithm that generates a graph of a link farm from an initial seed and propagates badness values through this graph. This information can then be used with common, link-based ranking algorithms, such as PageRank or HITS. The same authors also present their findings on cloaking and redirection techniques [29]. Ntoulas et al. [18] present a technique of detecting spam pages by content analysis. This work only takes query independent features into account, while Svore et al. [26] also use query dependent information. A system to detect cloaking pages is proposed by Chellapilla and Chickering in [5]. For this, a given URL is downloaded twice, providing different user agent strings for each download. If the pages are (significantly) different, the page uses cloaking techniques.

Wang et al. [27] follow the money in advertising schemes and propose a five-layer, double-funnel model to explain the relations that exist between advertisers and sites that employ web spam techniques. Fetterly et al. [7] present a series of measurements to evaluate the effectiveness in web spam detection. A quantitative study of forum spamming was presented by Niu et al. [17]

The work that is closest to our attempt in inferring the importance of different web page features is [2]. In that paper, Bifet et al. attempt to infer the importance of page features for the ranking algorithm by analyzing the results for different queries. They extract feature vectors for each page and try to model the ranking function by using support vector machines. Since their work is based on already existing pages, they do not have control over certain features (e.g., in-link properties). In [6], Evans performs a statistical analysis of the effect that certain factors have on the ranking of pages. While he includes factors, such as the listing of pages in web directories and a site's PageRank value, Evans only focuses on query independent values while neglecting all other factors.

6 Conclusions

Search engines are a target for attackers that aim to distribute malicious content on their websites or earn undeserved (advertising) revenue. This observation motivated our work to create a classifier that is able to identify and remove unwanted entries from search results. As a first step, we required to understand which features are important for the rank of a page. The reason is that these features are most likely the ones that an attacker will tamper with. To infer important features, we conducted an experiment in which we monitored, for almost three months, the ranking of pages

with 30 different combinations of feature values. Then, we computed the weights for the features that would best predict the actual, observed rankings. Those features with the highest weights are considered to be the most important for the search engine ranking algorithm. Based on the features determined in the first step and a labeled training set, we generated a classifier (a J48 decision tree). This decision tree was then evaluated on a test data set. The initial evaluation resulted in 35% detection rate and 11% false positives. By taking into account the confidence values of the decision tree and introducing a cutoff value, the false positives could be lowered to zero. At this rate, almost one out of five spam pages can be detected, improving the results of search engines without removing any valid results.

Acknowledgments This work has been supported by the Austrian Science Foundation (FWF) under grant P18764, SECoverer FIT-IT Trust in IT-Systems 2. Call, Austria, Secure Business Austria (SBA), and the WOMBAT and FORWARD projects funded by the European Commission in the 7th Framework.

Appendix A: J48 decision tree

See Fig. 2.

Appendix B: List of experiments

Since instances within an experiment group share the same feature values, only the experiment groups are listed here (Table 4).

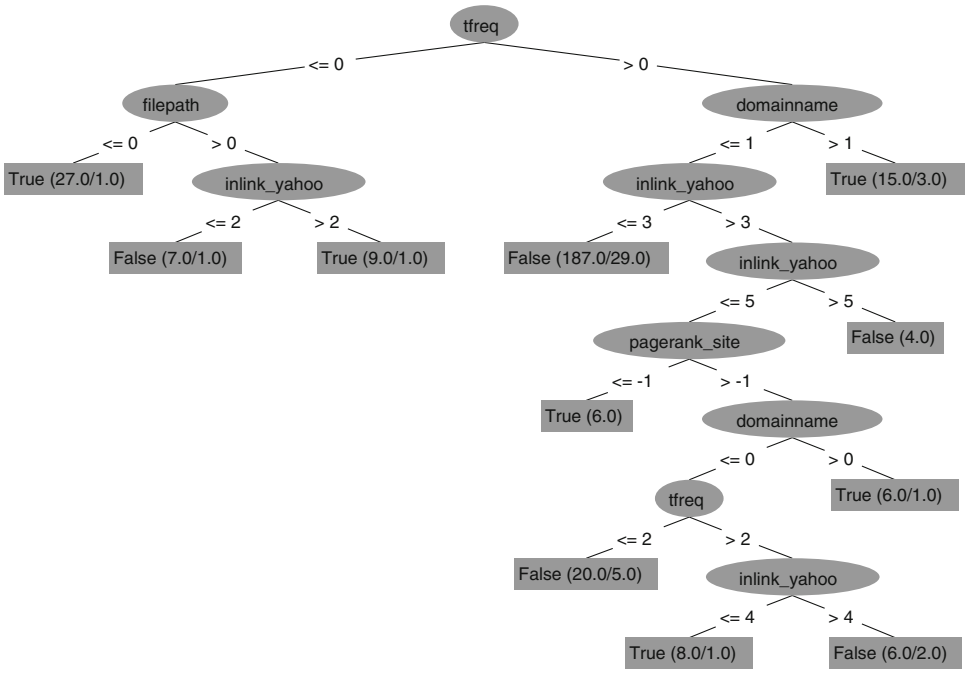

Fig. 2 Generated J48 decision tree. The node labels correspond to the feature extractors listed in Sect. 4.1

Table 4 List of experiment groups

No.	Feature combination	Description
1	1,2,3,4,7,9	Baseline
2	1,2,3,7,$9	Baseline with much text
3	1,2,3,$6,7,$9	Baseline with much text and many links to low quality sites
4	1,+2,3,7,9	Elevated use of keywords in `BODY`
5	1,$2,3,7,9	Keyword spamming of `BODY`
6	+1,2,3,7,9	Elevated use of keywords in the `TITLE`
7	$1,2,3,7,9	Keyword spamming of `TITLE`
8	1,2,3,$4,7,9,10	Keyword spamming of the URL
9	$1,$2,$3,$4,$5,7,9	Spam all on site
10	$1,$2,$3,$4,$5,$7,9	Spam all
11	$1,$2,$3,$4,$5,$7,$9	Spam all with much text
12	1,2,3,4,5,7,9	Include links to high quality pages

Table 4 continued

No.	Feature combination	Description
13	1,2,3,4,+5,7,9	Include more links to high quality pages
14	1,2,3,4,$5,7,9	Include many links to high quality pages
15	1,2,3,4,6,7,9	Include links to low quality pages
16	1,2,3,4,+6,7,9	Include more links to low quality pages
17	1,2,3,4,$6,7,9	Include many links to low quality pages
18	1,2,3,4,7,8,9	In-links with keywords in anchor text
19	1,2,3,4,7,9	In-links without keywords in anchor text
20	1,2,3,4,+7,8,9	Elevated amount of in-links with keywords in anchor text
21	1,2,3,4,+7,9	Elevated amount of in-links without keywords in anchor text
22	1,2,3,4,$7,8,9	Spam amount of in-links with keywords in anchor text
23	1,2,3,4,$7,9	Spam amount of in-links without keywords in anchor text
24	1,2,3,$4,7,9	URL keyword spam without domain name
25	1,2,3,4,7,9,10	Baseline with keyword in domain name
26	$1,$2,$3,$4,$5,$7,$9,10	Spam all with keyword in domain name
27	1,2,3,4,7,8,9	In-links with keywords and keywords in file name
28	1,2,3,4,7,9	In-links without keywords and keywords in file name
29	1,2,3,4,7,8,9,10	In-links with keywords and keywords in domain name
30	1,2,3,4,7,9,10	In-links without keywords and keywords in domain name

Column 2 references the features in Table 1 and captures the list of applied features for this experiment group. The lack of a feature in the description denotes that the feature is not used for this experiment, the prefix (+) indicates that a feature is applied in elevated quantities, where ($) means the feature is present in spam quantities. The third column is a description of the case that this experiment group reflects

References

1. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning. Artif. Intell. Rev. **11**(1–5), 11–73 (1997)
2. Bifet, A., Castillo, C., Chirita, P.-A., Weber, I.: An analysis of factors used in search engine ranking. In: Adversarial Information Retrieval on the Web (2005)
3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: 7th International World Wide Web Conference (WWW) (1998)
4. Cacheda, F., Viña, Á.: Experiencies retrieving information in the world wide web. In: Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC 2001), pp. 72–79 (2001)
5. Chellapilla, K., Chickering, D.: Improving cloaking detection using search query popularity and monetizability. In: Adversarial Information Retrieval on the Web (2006)
6. Evans, M.P.: Analysing Google rankings through search engine optimization data. Internet Res. **17**(1), 21–37 (2007)
7. Fetterly, D., Manasse, M., Najork, M.: Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In: WebDB, pp. 1–6 (2004)
8. Google. Zeitgeist: Search patterns, trends, and surprises. http://www.google.com/press/zeitgeist.html. Accessed 29 June 2009
9. Google Keeps Tweaking Its Search Engine. http://www.nytimes.com/2007/06/03/business/yourmoney/03google.html?page%wanted=4&_r=1. Accessed 29 June 2009
10. Gyöngyi, Z., Garcia-Molina, H.: Web spam taxonomy. In: Adversarial Information Retrieval on the Web (2005)
11. Hearst, M.A.: Support vector machines. IEEE Intell. Syst. **13**(4), 18–28 (1998)
12. Heckerman, D.: A tutorial on learning with bayesian networks. Technical report, Microsoft Research (1995)
13. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, August 18–20, 1995, Montreal, Quebec, Canada, pp. 338–345 (1995)
14. Kaburlasos, V.G., Athanasiadis, I.N., Mitkas, P.A.: Fuzzy lattice reasoning (flr) classifier and its application for ambient ozone estimation. Int. J. Approx. Reason. **45**(1), 152–188 (2007)
15. Karlberger, C., Bayler, G., Kruegel, C., Kirda, E.: Exploiting redundancy in natural language to penetrate bayesian spam filters. In: First USENIX Workshop on Offensive Technologies (WOOT07) (2007)
16. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
17. Niu, Y., Wang, Y.-M., Chen, H., Ma, M., Hsu, F.: A quantitative study of forum spamming using context-based analysis. In: NDSS (2007)
18. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: 15th International World Wide Web Conference (WWW) (2006)
19. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods: support vector learning, pp. 185–208. MIT Press, Cambridge. ISBN:0-262-19416-3 (1999)
20. Provos, N., Mavrommatis, P., Rajab, M.A., Monrose, F.: All your iframes point to us. In: 17th USENIX Security Symposium (2008)
21. Provos, N., McNamee, D., Mavrommatis, P., Wang, K., Modadugu, N.: The ghost in the browser analysis of web-based malware. In: First Workshop on Hot Topics in Understanding Botnets (HotBots '07) (2007)

22. Quinlan, R.: C4.5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)
23. Rahul Mohandas (McAfee Avert Labs). Analysis of Adversarial Code: The role of Malware Kits! http://clubhack.com/2007/files/Rahul-Analysis_of_Adversarial_Code.pdf, December 2007. Accessed 29 June 2009
24. Google Search Engine Ranking Factors. http://www.seomoz.org/article/search-ranking-factors. Accessed 29 June 2009
25. Shi, H.: Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton, NZ, COMP594 (2007)
26. Svore, K., Wu, Q., Burges, C., Raman, A.: Improving web spam classification using rank-time features. In: Adversarial Information Retrieval on the Web (2007)
27. Wang, Y.-M., Ma, M., Niu, Y., Chen, H.: Spam double-funnel: connecting web spammers with advertisers. In: 16th International Conference on World Wide Web (2007)
28. Witten, I., Frank, E.: Data mining: practical machine learning tools and techniques 2nd edn. Morgan Kaufmann, San Francisco (2005)
29. Wu, B., Davison, B.: Cloaking and redirection: a preliminary study. In: Adversarial Information Retrieval on the Web (2005)
30. Wu, B., Davison, B.D.: Identifying link farm spam pages. In: 14th International World Wide Web Conference (WWW) (2005)