Detecting Semantic Cloaking on the Web

Baoning Wu and Brian D. Davison
Department of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015 USA
{baw4,davison}@cse.lehigh.edu

ABSTRACT

By supplying different versions of a web page to search engines and to browsers, a content provider attempts to cloak the real content from the view of the search engine. Semantic cloaking refers to differences in meaning between pages which have the effect of deceiving search engine ranking algorithms. In this paper, we propose an automated two-step method to detect semantic cloaking pages based on different copies of the same page downloaded by a web crawler and a web browser. The first step is a filtering step, which generates a candidate list of semantic cloaking pages. In the second step, a classifier is used to detect semantic cloaking pages from the candidates generated by the filtering step. Experiments on manually labeled data sets show that we can generate a classifier with a precision of 93% and a recall of 85%. We apply our approach to links from the dmoz Open Directory Project and estimate that more than 50,000 of these pages employ semantic cloaking.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

Web search engine, spam

1. INTRODUCTION

Imagine the scenario in which a query is sent to a search engine but the top ranking pages are not relevant at all. This is quite disappointing. How could this happen? Cloaking, one type of search engine spamming technique, is a possible reason.

Users trust search engines in the sense that they expect that only the most relevant responses will be listed in the top ranking positions, and thus typically view just one page of results [35]. Since increased traffic to a commercial web site may bring more profit, more sites will compete for the top rankings, especially for popular queries.

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

Not all content providers will work hard to generate high quality web pages in order to get higher rankings. Some will find shortcuts and instead aim to make their pages rank highly by manipulating Web page features on which search engines' ranking algorithms are based. This behavior is usually called "search engine spam" [33, 19]. Henzinger et al. [22] have pointed out that search engine spam is one of the challenges that search engines need to face and win. Search engine results will be seriously impacted if no action is taken to detect spam.

A search engine generally ranks a page based on its content and linkage information. Ranking algorithms depend on the implicit assumption that the page content is real; i.e., the content seen by the search engines is identical to the content seen by actual users with browsers. But this assumption does not hold in the case of a significant web spamming technique called cloaking. By supplying different versions of a web page to search engines and to browsers, a content provider attempts to hide (cloak) the real content from the view of the search engine.

As a simple example, from a user's perspective through a web browser, a page may be a product page selling iPods, but the content sent to a search engine crawler might additionally contain brand names of digital cameras, laptop computers and camcorders. Furthermore, some keywords such as "cheap plane ticket", "free travel" might also be added.

In addition to other spamming techniques, major search engines typically declare their opposition to cloaking [17, 41, 4].

Surprisingly, some cloaking behavior is actually acceptable to, if not welcomed by, search engines [19]. For example, some content providers will not send advertising links to search engines or may generate PHP pages that do not contain session identifiers (that would otherwise be sent to users). To help distinguish between acceptable and unacceptable cloaking behaviors, we have defined two different types of cloaking [38]: syntactic and semantic cloaking. Syntactic cloaking includes all situations in which different content is sent to a crawler versus real users. Semantic cloaking is the subset of syntactic cloaking in which differences in meaning between pages are likely to deceive a search engine's ranking algorithm. Some experts name semantic cloaking as "malicious cloaking". Obviously, semantic cloaking is the behavior of interest to search engine providers.

Building on our previous experience [38] in detecting syntactic cloaking, we address in this paper the harder problem of how to recognize semantic cloaking automatically. We

propose a two-step process to detect semantic cloaking on the Web. The filtering step will detect all candidates that may utilize semantic cloaking. A classifier will then determine whether each candidate generated from the filtering step incorporates semantic cloaking or not.

This is the first publication that attempts to detect semantic cloaking automatically. While some ideas have been proposed to detect cloaking [38, 29], they may need human judgment to decide whether the pages employ semantic cloaking. The main contribution of this paper is the design and evaluation of a novel two-step process to identify semantic cloaking behavior on the web without human involvement. In experimental tests, our classifier achieves a precision of more than 90% and a recall of more than 80%.

The rest of this paper is organized as follows: related work will be introduced in Section 2. The motivation for detecting semantic cloaking will be discussed in Section 3. The details of our algorithms for detecting semantic cloaking are given in Section 4. Experimental work and results will be presented in Section 5. We conclude with a discussion and future work in Section 6.

2. RELATED WORK

Most search engine spamming techniques can be put into three different categories: content spam, link spam and page-hiding spam. Cloaking is the major technique within the page-hiding category. Several different methods have been proposed [36, 8, 16, 42, 6, 26, 10, 20, 13, 1, 28, 39, 14] to address content spam and link spam, but few publications address the cloaking issue.

Gyöngyi and Garcia-Molina [19] describe cloaking and redirection as spam hiding techniques. They noted that web sites can identify search engine crawlers by their network IP address or user-agent names. They additionally point out that some cloaking (such as sending the search engine a version free of navigational links and advertisements but no change to the content) is accepted by some engines. No solution for how to combat cloaking is introduced.

Perkins [33] argues that any agent-based cloaking is spam. No matter what kind of content is sent to search engine, the goal is to manipulate search engines rankings, which is an obvious characteristic of search engine spam. Again, no solution is given in this paper.

Cafarella and Cutting [8] consider cloaking as one of the spamming techniques. They said that search engines will fight cloaking by penalizing these sites. But detecting this cloaking behavior remains a problem.

Najork was awarded a patent [29] for a method of detecting cloaked pages. He proposed the idea of detecting cloaked pages from users' browsers by installing a toolbar and letting the toolbar send the signature of user perceived pages to search engines. His method may still have difficulty in distinguishing rapidly changing or dynamically generated Web pages from real cloaking pages, and does not directly address semantic cloaking.

Henzinger et al. [22] pointed out that search engine spam is quite prevalent and search engine results would suffer greatly without taking measures. They consider cloaking as one of the major search engine spam techniques. They suggest that crawling the same page twice may detect cloaking, but they also admit that this method has difficulty in differentiating legitimate changes, such as dynamic news headlines, from cloaking behavior.

In a workshop paper [38], we introduced the idea of automatic detection of cloaking pages using more than two copies of the page. We differentiated semantic cloaking from strictly syntactic cloaking, and explored methods for syntactic cloaking recognition.

Since we use machine learning methods to detect semantic cloaking, research using machine learning methods to detect spam are also relevant.

Amitay et al. [3] propose categorization algorithms to detect website functionality. They recognized that the sites with similar roles exhibit similar structural patterns. After finding one special pattern for a certain kind of website, they can predict the websites to be in the same class if the sites show a similar pattern. For each host, they selected 16 features such as the average level of the page, in-links per page, out-links per leaf page, etc., to do the categorization. In their experimental results, they claimed to have identified 31 clusters, each of which appears to be a spam ring.

"Nepotistic links", i.e., those links between pages that are present for reasons other than merit, are bad for the link-based ranking algorithm, so it is necessary to get rid of them. Davison [13] uses multiple features of web pages to detect nepotistic links. Features included page title, page description, initial IP address octets, common outgoing links, etc.

Recently, Drost and Scheffer [14] proposed using machine learning methods to detect link spam. They generated 89 features for each page and manually labeled hundreds of examples of link spam. These examples, along with non-spam examples from the Open Directory Project [31], were used to train a classifier to recognize link spam pages.

3. MOTIVATION

Cloaking occurs when different content for the same URL is sent to browsers and to search engine crawlers. Since page content is one of the most important components in ranking pages, major search engines typically discourage cloaking [17, 41, 4].

The assumption made by search engines is that anyone utilizing cloaking has the potential to fool search engines to generate a higher ranking for their pages. However, based on our experience, this is not always the case.

In previous work [38], we detected that about 9% of pages within a hot query data set utilized cloaking. (We will introduce this data in Section 5.) The ratio is high enough to show that cloaking is not scarce on the Web. After manually checking approximately 1,000 examples of cloaking, we found that some of them will affect search engine results while others are not likely to be harmful to search engines.

For example, we found that many web servers that use PHP or ASP to generate dynamic content will have session identifiers within URLs for copies sent to browser, but no such session identifiers are sent to a crawler. This is reasonable in the sense that the Web servers need this kind of session identifier to differentiate different users, but these identifiers are not useful to search engines. If these identifiers were sent to search engines, search engines might think that the page has changed on every visit while actually the page has not.

Another example is that some pages will contain advertising links when being viewed from a browser, but no such links are present in the copy sent to crawlers. In fact, this behavior is helpful for search engines utilizing link-based ranking algorithms. The reason is that the base assump-

tion of link-based ranking algorithms is that a link on the Web implicitly represents an authority vote [7, 25] or a kind of trust [20]. Obviously, advertising links don't match this assumption. Hence, this kind of cloaking can help search engines collect fewer advertising links, thus generating a better ranking.

Based on the above observations, we find that it is not enough to detect syntactic cloaking behavior. If search engines penalize all instances of cloaking, obviously it is not fair to the above examples. However, given the enormous size of the Web, it is also not possible for human experts to check each page manually to decide which one to penalize. Hence, it is necessary to find a method to detect the kind of cloaking behavior that has the effect of manipulating search engine ranking results. We refer to this cloaking behavior as semantic cloaking.

4. ALGORITHM

In order to decide whether a Web page is employing cloaking, copies from both a browser's perspective and a crawler's perspective are needed [22]; a single copy in the search engine's cache is not enough. The reason is that spammers may send high quality content (e.g., copied from Yahoo!, CNN or Wikipedia) to the search engine's crawler; therefore, flagging content-based features of the crawled version alone is insufficient.

Also, the comparison of only one copy of a page from a browser's perspective and only one copy from a crawler's perspective is still insufficient for detecting cloaking [38, 22]. For example, a college's homepage may highlight a different faculty members' biographical information every time the page is served. It is not easy to discover this behavior by only looking at one copy from both browser's perspective and crawler's perspective. So, two or more copies from each side are necessary for detecting cloaking. This is also true in the case of detecting semantic cloaking.

The problem with this 4-copy method is that it is expensive for search engines. With billions of pages on the Web today, crawling and storing four copies and calculating the differences among them requires significant resources. Hence, we propose a two-step process for detecting semantic cloaking requiring significantly fewer resources.

The first step implements a filter. By comparing only one copy from the browser's perspective and the crawler's perspective, heuristics are used to eliminate Web pages that cannot demonstrate cloaking. The goal of this step is to generate a candidate list with a reasonable precision but a perfect recall for semantic cloaking pages. All the pages that are not eliminated in the first step are sent to the second step for further inspection.

In the second step, first two more copies, one from a browser's perspective and one from a crawler's perspective, are downloaded for each candidate. Then features are extracted from across these four copies and a classifier is used to determine whether this page is doing semantic cloaking or not. The overall process is depicted in Figure 1.

4.1 Filtering Step

The purpose of this step is to eliminate pages that do not employ semantic cloaking.

The motivation for this step is simple: if we can tell that the page is not utilizing cloaking at all with a smaller cost, then there is no need for us to download four copies for this page for further inspection. For example, if the copy from a browser's perspective and from a crawler's perspective are identical, the chance is very small that this page is employing semantic cloaking.

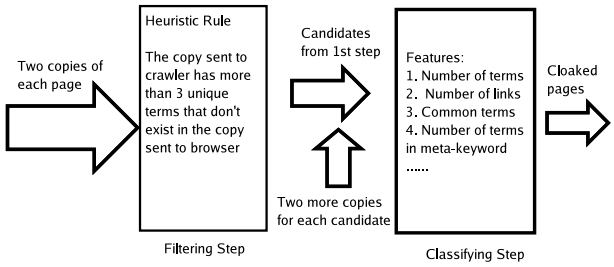


Figure 1: Data flow of the two steps for detecting semantic cloaking.

One possible direct rule for the filtering step is to eliminate all pages for whom the browser and crawler perspectives are identical, and to send all the pages that are different in the two copies to the classification step. Although this is a safe rule for the first step, we find this rule is still insufficient. The reason is that the Web is changing very fast. Cho and Garcia-Molina [12] showed that about 20% of Web pages changed every time they were visited and about 40% of Web pages were changed within a week. Other researchers [15, 30] also show that Web pages are changing fast. This shows that by applying the above "identical rule", we may still get a lot of candidates for the classification step.

As discussed before, even if the copy from a browser's perspective and the copy from a crawler's perspective are different, it still doesn't necessarily mean that the page is utilizing semantic cloaking. For example, some servers will send JavaScript to the browser but not to the crawler. Hence, a better heuristic for eliminating pages that are not likely to perform semantic cloaking is needed.

In order to determine a better heuristic rule, we carefully study the manually detected cloaking examples. One common characteristic of these examples is that spammers are ambitious. They will often list more keywords or links in the copy sent to the crawler than the copy sent to the browser if they utilize cloaking. Typically these keywords are real words for popular queries, e.g., names of hot games, pop stars, digital cameras. An example of a portion of a semantic cloaking page with many keywords is shown in Figure 2.

Hence, the ideal situation is that there exists a complete enough dictionary that contains all the keywords that may be used for spamming purpose. Thus, the heuristic rule can be that if the copy sent to the crawler contains a number of dictionary terms that don't exist in the copy sent to the browser, then this page will be marked as a candidate and sent to the classification step.

A threshold can be used in which pages with more than this threshold of words only in the copy sent to the crawler or sent to the browser will be marked as candidates. We refer to this heuristic rule as the "term rule" in this paper. Different thresholds will generate different candidate lists. Obviously, one is the optimal threshold in the sense that no semantic cloaking page will be filtered out during this filtering step.

As we have described above, spammers may also put dif-

game info, reviews, game reviews, previews, game previews, interviews, features, articles, feature articles, game developers, developers, developer diaries, strategy guides, game strategy, screenshots, screen shots, game screenshots, game screen shots, screens, forums, message boards, game forums, cheats, game cheats, cheat codes, playstation, playstation, dreamcast, Xbox, GameCube, game cube, gba, game, advance, software, game software, gaming software, files, game files, demos, game demos, play games, play games online, game release dates, Fargo, Daily Victim, Dork Tower, classics games, rpg, role playing games, strategy games, real time strategy, action games, FPS, shooter, first person shooters, sports games, handheld games, pda, pda games, half-life, half life, quake, quake, counter strike, counter strike, rogue spear, rainbow six, medal of honor, allied assault, unreal tournament, return to castle Wolfenstein, soldier of fortune, tribes, starsiege tribes, command and conquer, renegade, halo, grand theft auto, tony hawk, the sims, sims

Figure 2: Sample of a meta-keyword tag sent in both HTTP response header and HTML content only to the crawler.

ferent links in the copy sent to the crawler and in the copy sent to the browser. Similarly, we can have a rule for links: first we extract all links from pages and if a page has more than a threshold of unique links that don't occur in the copy sent to the browser or do not occur in the copy sent to the crawler, we will mark this page as a candidate and send it to the classification step. We refer to this rule as the "link rule". Again, the optimal threshold will be one for this link rule. Any page that matches either term rule or link rule will be marked and sent to the second step.

4.2 Classification Step

The filtering step above will admit candidate pages. We have also explained that a single copy each from the crawler and the browser is insufficient to detect semantic cloaking. Hence, for each candidate, we download another copy from both browser and crawler perspectives. For consistency, we download these two copies in the same order as the one for the initial two copies, i.e., if we use B_1, B_2, and C_1, C_2 to represent the copies downloaded as a browser and a crawler respectively, then the downloading sequence of these four copies should be either $B_1{-}C_1{-}B_2{-}C_2$ or $C_1{-}B_1{-}C_2{-}B_2$.

From the four copies of a page, a set of features can be generated. We use a set of labeled pages for training a classifier to determine whether or not the page is employing semantic cloaking.

A major concern is how to generate a set of useful features. As in most machine learning tasks, this is best performed using domain expertise. Based on our examination of many cloaking and non-cloaking pages, and with raw data in the form of four parsed versions of a page in mind, we have created a number of potentially useful features.

One goal for our classifier is widespread applicability to unseen pages — as a result, where possible we choose features that are independent of the actual content of the pages. Most features represent information about the content, rather than the content directly. Given that we are examining hypertext, we can extract features based both on the content (text, markup, etc.) as well as links to other

Content-based features from each copy

Response code.

Number of

- terms.
- terms in the title field of HTTP response header.
- terms in the keyword field of HTTP response header.
- terms in the description field of HTTP response header.
- lines in HTTP response header.

Whether

- "method=post" exists.
- "input type=hidden" exists.
- Google AdSense exists.
- checkbox exists.
- radiobox exists.
- "type=submit" exists.
- JavaScript exists.
- frame tag exists.
- "onmouseover" function exists.
- stylesheet exists.
- a reference to a flash file exists.
- copyright information exists.
- a reference to a JavaScript file exists.

Figure 3: Content-based features from each copy.

Link-based features from each copy

Number of

- total links.
- unique links.
- links to the same site.
- links to a different site.
- relative links.
- absolute links.
- unique sites that this page points to.

Ratio of number of

- links to the same site to the number of total links.
- links to a different site to the number of total links.
- absolute links to the number of total links.
- relative links to the number of total links.
- links to the same site to the number of links to a different site.
- absolute links to the number of relative links.

Figure 4: Link-based features from each copy.

pages. Note that although "title", "keyword" and "description" fields are not a part of the standard HTTP response header, we observe that many spammers include these fields into the HTTP response header. Hence, we take this observation into account in our feature selection process. We show content and link-based features of one instance of a

page in Figures 3 and 4, respectively.

Since at this point we have four copies of each page, we can also consider the differences across and within user-agents. Intuitively, such differences are important as they can distinguish between time-changing content versus static differences between versions sent to the two user-agents. We generate features for both kinds of differences.

Therefore, given two instances, we compare the feature values for many features (e.g., is the number of links contained in each the same), and we compare aspects of the content (e.g., how many terms appear only one instance, but not the other). These features are calculated from the comparison of instances sent to the same client (e.g., from B_1 to B_2), and from the comparison of instances retrieved first or last (e.g., from C_1 to B_1). Some examples of these features are shown in Figure 5.

By combining the various features described above, we finally determine 162 features for each candidate page to train the classifier.

4.3 Our Implementation

In reality, the ideal approach described in Section 4.1 can not be easily implemented. For example, it is quite difficult to find a powerful dictionary to contain all valid words and names (although a search engine might be able to construct something close). Hence we employ a non-optimal heuristic rule in our implementation. For the term rule, we decompose all the pages into terms by replacing all the non-alphanumeric characters with blanks. We then drop all the terms containing digits. For the remaining terms, if the copy sent to the crawler contains more than a threshold of terms that don't exist in the copy sent to the browser, we mark it as a candidate and send it to the second step.

For the link rule, we extract all the links within the "HREF" tags for all the pages. If the copy sent to the crawler contains more than a threshold of links that don't exist in the copy sent to the browser, we mark it as a candidate and send it to the second step.

For the second step, we employ a support vector machine as our classifier model. The software SVM^{light} [23] is used to train the SVM classifier.

5. EXPERIMENTS

In this section, we first show how we build an SVM classifier and its performance based on a small data set. We then apply our two-step process to detect semantic cloaking based on a large data set.

5.1 Building the classifier

The data set used for training the classifier is the set of URLs included in top response pages for hot queries. This data set is the same set used our prior work in detecting cloaking [38].

5.1.1 Data set

We collected ten popular queries of Jan. 2005 from Google Zeitgeist [18], the top 100 search terms of 2004 from Lycos [27], the top ten searches for the week ending Mar. 11, 2005 from Ask Jeeves [5], and ten hot searches in each of 16 categories ending Mar. 11, 2005 from AOL [2]. This resulted in 257 unique queries from these web sites.

Features for two corresponding copies

Whether

- the number of terms in the keyword field of the HTTP response header for C_1 is bigger than the one for B_1.
- the number of terms in the keyword field of the HTTP response header for C_2 is bigger than the one for B_2.
- the total number of links in B_1 is the same as the total number of links in B_2.
- the total number of links in C_1 is the same as the total number of links in C_2.

Number of

- common terms in B_1 and C_1.
- common links in B_1 and C_1.
- terms appearing only in B_1, not in C_1.
- links appearing only in B_1, not in C_1.

Figure 5: Features for two corresponding copies

For each of these queries, we retrieved the top 200 responses from the Google search engine. The number of unique URLs is 47,170. In each case we record the entire response – the HTTP headers, including response code, plus the body of the response. We also record, and follow, HTTP redirection responses.

When a client retrieves a resource from a Web server, it typically includes a `UserAgent` header field to identify the type of client. In order to disguise our request as either a search engine robot or a normal web browser, we changed the `UserAgent` HTTP header when we sent out our requests. To pretend to be a search engine robot, we set the `UserAgent` to be `Googlebot/2.1 (+http://www.googlebot.com/bot.html)`. To pretend to be a web browser, we set the `UserAgent` to be `Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)`. We then downloaded four copies for each of the 47,170 URLs — two from a browser's perspective and two from a crawler's perspective. However, unlike real crawlers, all of these retrievals were performed using machines with a university IP address, meaning that pages performing cloaking based solely on IP address will be missed.

5.1.2 Labeled data set

In order to train a classifier, we need a labeled data set containing both positive and negative examples of semantic cloaking from the hot query data set. We used the following procedure to get such a list.

1. For the 4,083 pages identified as syntactic cloaking in [38], select one URL for each unique site.
2. For the list generated from the above step, manually check whether the URL is doing semantic cloaking.
3. For each of the URLs that are marked as semantic cloaking in step 2, extract the site and select all the pages from this site within the hot query data set.
4. For the list generated from step 3, double check whether the URL is doing semantic cloaking.
5. Combine the list from step 2 and step 4 to form the labeled list.

Top positive features for detecting semantic cloaking

1. Whether the number of terms in the keyword field of the HTTP response header for C_1 is bigger than the one for B_1.
2. Whether the number of terms in the keyword field of the HTTP response header for C_2 is bigger than the one for B_2.
3. Whether the number of terms in the description field of the HTTP response header for C_1 is bigger than the one for B_1.
4. Whether the number of terms in the description field of the HTTP response header for C_2 is bigger than the one for B_2.
5. Whether the number of lines in the HTTP response header for C_1 is bigger than the one for B_1.
6. Whether the number of terms in the title field of the HTTP response header for C_1 is bigger than the one for B_1.
7. Whether the number of terms in the title field of the HTTP response header for C_2 is bigger than the one for B_2.
8. Whether the number of unique terms in C_1 is bigger than the one in B_1.
9. Whether C_1 has the same number of relative links as B_1.
10. Whether the number of common terms between C_1 and B_1 is same as the number of common terms between C_2 and B_2.

Figure 6: Top 10 features with positive weights.

Following the above procedure, we get a labeled list of 1,285 URLs. Among them, 539 are positive examples and 746 are negative examples. Here positive means the page is doing semantic cloaking.

5.1.3 Training the classifier

We partitioned the labeled list into a training list and a test list. The training list contains randomly selected 60% of both positive and negative examples; the remaining 40% of both positive and negative examples are put into the test list. This ends up with a training set of 767 examples and a test set of 518 examples.

For these 1,285 pages, we extracted 162 features from the four copies and used SVM^{light} [23] to build a classifier. In order to prevent features with large values (such as the number of tokens within one copy) from dominating the classifier, we scale down all features to the range [0,1]. After this scaling, we train the classifier and use precision of and recall as the metrics. Here precision means what percentage of the pages predicted by the classifier as semantic cloaking pages are actually utilizing semantic cloaking. Recall refers to the fraction of all semantic cloaking pages that are detected by the classifier. A model with 278 representative feature vectors is built by SVM^{light} after the training process.

In order to avoiding overfitting, we repeated the training set and test set selection process five times. The average precision of these five runs is 93% and recall is 85%. This demonstrates that using the classifier to detect semantic cloaking is promising.

Top negative features for detecting semantic cloaking

1. Whether C_1 uses the same stylesheet file as B_1.
2. Whether C_2 uses the same stylesheet file as B_2.
3. Whether both C_2 and B_2 contain copyright information.
4. Whether both C_2 and B_2 contain JavaScript.
5. Whether both C_2 and B_2 contain "METHOD POST".
6. Whether both C_1 and B_1 contain JavaScript.
7. Whether C_1 has the same copyright information as B_1.
8. Whether both C_1 and B_1 contain "METHOD POST".
9. Whether both C_2 and B_2 contain checkbox.
10. Whether B_1 and B_2 have same number of terms in the META DESCRIPTION field.

Figure 7: Top 10 features with negative weights.

5.1.4 Discriminative features

We have 162 features in total. Obviously not all of them are of the same importance to the classifier. In order to know which features are most discriminative, we use two methods.

First, like Drost and Scheffer [14], we calculate the weight for each feature and output the features with highest weights as the most discriminative features. We used the model file generated by SVM^{light} to calculate the weight for each feature. The top ten features with a positive weight are shown in Figure 6. These features can be considered as the most discriminative features to tell that the page is utilizing semantic cloaking.

Many features in Figure 6 are related to the HTTP response header. The reason is that we detect many semantic cloaking instances using HTTP response headers to deliver spam. Many keywords are added to the HTTP header sent to the crawler.

Similarly, the top ten features with negative weights are shown in Figure 7. These features are important to indicate that a page is not employing semantic cloaking.

The second method to present important features is to generate a decision tree. We use the WEKA [37] implementation of C4.5 [34] to generate a decision tree for our training data set, shown in Figure 8. The corresponding features for this tree are listed in Figure 9.

The features listed in Figure 9 include discriminative features for both positive and negative examples. Most of them are similar to the combination of features listed in Figure 6 and Figure 7. This demonstrates that the above features are discriminative features in deciding whether the page is utilizing semantic cloaking or not.

5.2 Detecting Semantic Cloaking

5.2.1 ODP data set

In order to demonstrate the value and validate performance of our two-step process of detecting semantic cloaking, we use a much larger data set.

Since the pages within dmoz Open Directory Project [31] are regularly used by web search, crawling, and classification

```
A <= 0
|   B <= 0.008406
|   |   C <= 0
|   |   |   D <= 0.000446: -1 (11.0/1.0)
|   |   |   D > 0.000446
|   |   |   |   E <= 0.000179: 1 (15.0)
|   |   |   |   E > 0.000179
|   |   |   |   |   F <= 0.007218: -1 (2.0)
|   |   |   |   |   F > 0.007218: 1 (2.0)
|   |   C > 0
|   |   |   G <= 0
|   |   |   |   H <= 0: 1 (10.0)
|   |   |   |   H > 0
|   |   |   |   |   I <= 0.005792
|   |   |   |   |   |   J <= 0: 1 (5.0/1.0)
|   |   |   |   |   |   J > 0: -1 (11.0)
|   |   |   |   |   I > 0.005792: 1 (2.0)
|   |   |   G > 0
|   |   |   |   K <= 0
|   |   |   |   |   L <= 0.9753: -1 (380.0/17.0)
|   |   |   |   |   L > 0.9753
|   |   |   |   |   |   M <= 0
|   |   |   |   |   |   |   N <= 0: -1 (4.0)
|   |   |   |   |   |   |   N > 0: 1 (8.0/1.0)
|   |   |   |   |   |   M > 0: -1 (11.0/1.0)
|   |   |   |   K > 0
|   |   |   |   |   O <= 0: 1 (4.0)
|   |   |   |   |   O > 0
|   |   |   |   |   |   P <= 0: 1 (3.0)
|   |   |   |   |   |   P > 0
|   |   |   |   |   |   |   Q <= 0.000141: 1 (2.0)
|   |   |   |   |   |   |   Q > 0.000141: -1 (37.0/3.0)
|   B > 0.008406
|   |   R <= 0
|   |   |   U <= 0: 1 (2.0)
|   |   |   U > 0: -1 (9.0/1.0)
|   |   R > 0
|   |   |   T <= 0
|   |   |   |   S <= 0: 1 (8.0)
|   |   |   |   S > 0
|   |   |   |   |   G <= 0: 1 (2.0)
|   |   |   |   |   G > 0: -1 (3.0)
|   |   |   T > 0: 1 (97.0/1.0)
A > 0: 1 (139.0)
```

Figure 8: Tree generated by the C4.5 algorithm.

research (e.g., [21, 9, 11]), it is prudent to see how many pages listed in the ODP utilize semantic cloaking.

We use the 2004 ODP RDF file [32] and extract 4,378,870 URLs from it. We then download two copies for each of these URLs, one pretending to be a crawler and one pretending to be a browser, for each of these URLs for our experiment.

5.2.2 Filtering step

As introduced in Section 4.1, the first step is to filter out the pages that may not employ semantic cloaking. As before, we first decompose all the pages into terms by replacing all the non-alphanumeric letters by blanks, collect the remaining terms and drop the terms containing digits.

Features selected by C4.5 for detecting semantic cloaking

- **A** Whether the number of terms in the description field of the HTTP response header for C_1 is bigger than the one for B_1.
- **B** The number of terms appearing only in C_2, not in B_2.
- **C** Whether C_1 uses the same stylesheet file as B_1.
- **D** The number of terms appearing only in C_1, not in B_1.
- **E** The ratio of the number of absolute links in C_1 to the number of relative links in C_1.
- **F** The number of unique terms in C_1.
- **G** Whether both C_2 and B_2 contain "METHOD POST".
- **H** Whether both C_1 and B_1 contain "input hidden".
- **I** The number of different terms between the title field of the HTTP response header for C_1 and the title field of the HTTP response header for B_1.
- **J** Whether both C_1 and B_1 contain a flash file.
- **K** Whether the number of relative links in C_1 is bigger than the number of relative links in B_1.
- **L** The ratio of the number of absolute links in C_2 to the total number of links in C_2.
- **M** Whether the total number of links in B_1 is the same as the total number of links in B_2.
- **N** Whether the number of different sites that C_1 points to is bigger than the number of different sites that C_2 points to.
- **O** Whether both C_2 and B_2 contain JavaScript.
- **P** Whether C_1 has the same copyright information as B_1.
- **Q** The number of terms appearing only in B_1, not in C_1.
- **R** Whether the total number of links in C_2 is bigger than the total number of links in B_2.
- **S** Whether both C_2 and B_2 contain checkbox.
- **T** Whether the number of terms appearing only in B_1 but not C_1 is the same as the number of terms appearing only in B_2 but not C_2.
- **U** Whether both C_1 and B_1 contain "METHOD POST".

Figure 9: Features selected by the C4.5 algorithm.

In order to investigate which threshold to choose for the term and link rules described in Section 4.1, we tried different thresholds from 1 to 10 for the term rule. The percentage of pages remaining for these different thresholds are shown in Figure 10. We arbitrarily choose 3 as our threshold for the term rule. The reason is that intuitively less than 3 different terms won't be a useful cloaking instance for spammers.

Since the candidate set generated by the link rule with threshold 3 is a subset of the set generated by the term rule, we use the candidate set generated by the term rule for the next step.

The filtering step marked 364,993 pages as cloaking candidates. That corresponds to 8.34% of the original set of pages. While a higher threshold could also be used, we believe that with better heuristic rules, it may be possible to filter out more candidates without the dropping of real semantic cloaking pages.

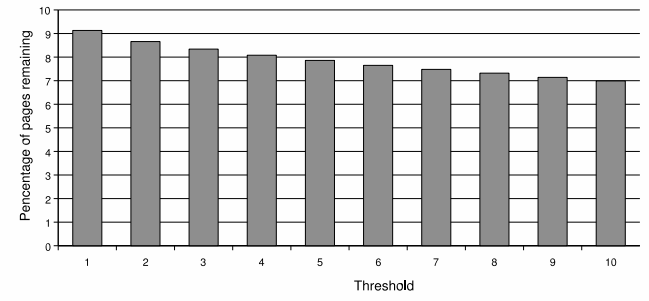


Figure 10: Percentage of pages remaining for different thresholds.

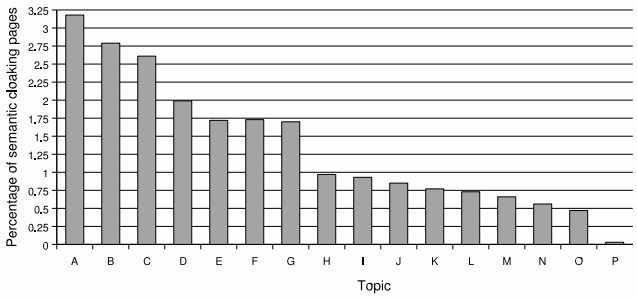
5.2.3 *Classification step*

For each of these 364,993 pages, two more copies, i.e., one from a crawler's perspective and one from a browser's perspective, are downloaded and then features are extracted from the four copies. Again, we use the same scale from training the model to scale down these features. Hence it is possible that absolute values of some features may not be within the range [0,1]. After classification, 46,806 pages are marked as utilizing semantic cloaking.

Since the classifier was trained on a different data set, we randomly select 400 pages from the set of 364,993 pages to estimate the performance of the classifier for this ODP data set. After manual checking, 52 semantic cloaking pages are detected. Then the classifier is applied to the above labeled 400 pages, the results are still good: accuracy is 96.8%, the precision is 91.5% and recall is 82.7%. Hence, this demonstrates that the classifier works well for the ODP data set.

Since the precision is 91.5% for the ODP data set and we get 46,806 pages classified as semantic cloaking, roughly speaking we have 46,806 * .915 = 42,827 true positive semantic cloaking instances in the ODP. If we also consider the cloaking in the false negatives, we get 42,827 / .827 = 51,786 cloaked pages in total in the ODP. Compared to the overall size of the ODP data set, i.e., about 4.3M pages, we see that more than 1% of all pages within the ODP are expected to utilize semantic cloaking. Importantly, since our estimates do not account for IP-based cloaking, actual cloaking rates in the ODP will be even higher.

To see whether cloaking is equally distributed across ODP topics, we calculate the distribution of the 46,806 marked pages within each of the 16 top level topics in the ODP. The results are shown in Figure 11. It makes some sense that popular topics such as "Games", "Recreation" and "Sports" also contain the highest fractions of semantic cloaking pages among all the topics. It is interesting to observe that the topic "Arts" also contains many semantic cloaking pages.



A. Arts
B. Games
C. Recreation
D. Sports
E. Home
F. Society
G. Kids&Teens
H. Computers
I. Health
J. Science
K. Regional
L. World
M. Shopping
N. Reference
O. Business
P. News

Figure 11: Percentage of semantic cloaking pages within each topic.

6. SUMMARY AND DISCUSSION

We have proposed a two-step process to address the problem of detecting semantic cloaking on the Web. The filtering step generates a candidate set with a reasonable precision but a high recall for detecting semantic cloaking. The classification step identifies semantic cloaking pages based on a pre-trained classifier and features from four instances of a page. When tested, precision and recall for the classifier show good performance, at 93% and 85%, respectively. We also show that many semantic cloaking pages exist in the ODP.

For the filtering step, we use a simple heuristic rule, i.e., if more than three non-digit-containing terms are present in the crawler's copy but not in the browser's copy, we will put the page into the candidate list. More work can be done for this step in the future. Different heuristic rules or different thresholds can be used to filter out more pages without employing semantic cloaking.

For the classifier, we think more features may be added later to improve the performance. For example, we focus on the terms in the title, description, keyword fields and the number of terms in total when training the classifier. We observe that in the ODP data set, a labeled page uses ALT tag to apply semantic cloaking and our classifier fails to recognize it. Hence, more features may be necessary to make the classifier more robust. For example, we may consider features related to the images within Web pages.

Also, during manual labeling, we found some pages to be difficult to classify as utilizing semantic cloaking or not. Therefore, it may be beneficial to incorporate a separate class for such pages, and thus could consider ordinal classification techniques [24, 40].

We detect cloaking by forging different `UserAgent`s. Thus, we will miss cloaking instances utilizing IP-based cloaking, i.e., cloaking behavior based on the IP address of the machines that send out HTTP requests. Hence, our collection is a lower bound of semantic cloaking instances. We fully expect search engines to observe a higher fraction of cloaking pages when retrieving Web pages from known IP addresses that are used for search engine crawlers.

Finally, what are the consequences of publicizing our approach to spammers? We argue that our method is robust as long as the spammers are ambitious. Spammers can only bypass the filtering step of our algorithm by listing very few additional keywords in the copy sent to the search engines.

Sophisticated spammers might also tune their pages to reduce their profile within the strong features that we have identified for the classifier presented in this paper. However, with many features, it will be difficult (and unlikely) for a spammer to avoid all such features. Regardless, periodic retraining of the classifier is likely to be necessary to maintain peak performance, which will make the identification of any particular features here moot. In conclusion, we believe our

approach will identify most semantic cloaking, and make undetected cloaking more difficult and/or less effective.

7. REFERENCES

[1] A. Acharya, M. Cutts, J. Dean, P. Haahr, M. Henzinger, U. Hoelzle, S. Lawrence, K. Pfleger, O. Sercinoglu, and S. Tong. Information retrieval based on historical data, Mar. 31 2005. US Patent Application number 20050071741.

[2] America Online, Inc. AOL Search: Hot searches, Mar. 2005. http://hot.aol.com/hot/hot.

[3] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: Detecting site functionality by structural patterns. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, pages 38–47, Aug 2003.

[4] AskJeeves / Teoma Site Submit managed by ineedhits.com: Program Terms, 2005. Online at http://ask.ineedhits.com/programterms.asp.

[5] Ask Jeeves, Inc. Ask Jeeves About, Mar. 2005. http://sp.ask.com/docs/about/jeevesiq.html.

[6] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 104–111, Melbourne, AU, 1998.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[8] M. Cafarella and D. Cutting. Building Nutch: Open source. *Queue*, 2(2):54–61, Apr. 2004.

[9] S. Chakrabarti, M. Joshi, K. Punera, and D. Pennock. The structure of broad topics on the web. In *Proceedings of 11th International World Wide Web Conference*, pages 251–262, Honolulu, Hawaii, US, 2002. ACM Press.

[10] S. Chakrabarti, M. Joshi, and V. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *Proceedings of the 24th Annual ACM SIGIR International Conference on Research & Development in Information Retrieval*, pages 208–216, 2001.

[11] P. Chirita, W. Nejdl, R. Paiu, and C. Kohlschutter. Using ODP metadata to personalize search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 178–185, Salvador, Brazil, August 2005.

[12] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the Twenty-sixth International Conference on Very Large Databases (VLDB)*, 2000.

[13] B. D. Davison. Recognizing nepotistic links on the Web. In *Artificial Intelligence for Web Search*, pages 23–28. AAAI Press, July 2000. Presented at the AAAI-2000 workshop on Artificial Intelligence for Web Search, Technical Report WS-00-01.

[14] I. Drost and T. Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proceedings of European Conference on Machine Learning*, pages 96–107, Oct. 2005.

[15] D. Fetterly, M. Manasse, and M. Najork. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International World Wide Web Conference*, pages 669–678, Budapest, Hungary, May 2003.

[16] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *Proceedings of WebDB*, pages 1–6, June 2004.

[17] Google, Inc. Google information for webmasters, 2005. Online at http://www.google.com/webmasters/faq.html.

[18] Google, Inc. Google Zeitgeist, Jan. 2005. http://www.google.com/press/zeitgeist/zeitgeist-jan05.html.

[19] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Chiba, Japan, 2005.

[20] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 271–279, Toronto, Canada, Sept. 2004.

[21] T. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 517–526, Honolulu, Hawaii, May 2002.

[22] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, Fall 2002.

[23] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[24] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[25] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[26] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1–6):387–401, 2000.

[27] Lycos. Lycos 50 with Dean: 2004 web's most wanted, Dec. 2004. http://50.lycos.com/121504.asp.

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

[29] M. Najork. System and method for identifying cloaked web servers, June 21 2005. U.S. Patent number 6,910,077.

[30] A. Ntoulas, J. Cho, and C. Olston. What's new on the web? The evolution of the web from a search engine perspective. In *Proceedings of 13th International World Wide Web Conference*, pages 1–12, New York City, USA, May 2004.

[31] Open Directory Project, 2005. http://dmoz.org/.

[32] Open Directory RDF Dump, 2005. http://rdf.dmoz.org/.

[33] A. Perkins. White paper: The classification of search engine spam, Sept. 2001. Online at http://www.silverdisc.co.uk/articles/spam-classification/.

[34] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman, San Mateo, CA, 1993.

[35] C. Silverstein, M. Henginger, J. Marais, and M. Moricz. Analysis of a very large AltaVista query log. *SIGIR Forum*, 33:6–12, 1999.

[36] A. Westbrook and R. Greene. Using semantic analysis to classify search engine spam, Dec. 2002. Class project report at http://www.stanford.edu/class/cs276a/projects/reports/.

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

[38] B. Wu and B. D. Davison. Cloaking and redirection: A preliminary study. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, May 2005.

[39] B. Wu and B. D. Davison. Identifying link farm spam pages. In *Proceedings of the 14th International World Wide Web Conference*, pages 820–829, Chiba, Japan, May 2005.

[40] J. Xu, Y. Cao, H. Li, and M. Zhao. Ranking definitions with supervised learning methods. In *Proceedings of the 14th International World Wide Web Conference*, pages 811–819, May 2005.

[41] Yahoo! Inc. Yahoo! Help - Yahoo! Search, 2005. Online at http://help.yahoo.com/help/us/ysearch/deletions/.

[42] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. V. Roy. Making eigenvector-based reputation systems robust to collusions. In *Proceedings of the Third Workshop on Algorithms and Models for the Web Graph*, Oct. 2004.