Expert Systems with Applications 36 (2009) 7493–7499

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Detection of cloaked web spam by using tag-based methods

Jun-Lin Lin *

Department of Information Management, Yuan Ze University, 135 Yuan-Tung Road, Chung-Li 32003, Taiwan

ARTICLE INFO

Keywords:
Cloaking detection
Web spam
Classification

ABSTRACT

Web spam attempts to influence search engine ranking algorithm in order to boost the rankings of specific web pages in search engine results. Cloaking is a widely adopted technique of concealing web spam by replying different content to search engines' crawlers from that displayed in a web browser. Previous work on cloaking detection is mainly based on the differences in terms and/or links between multiple copies of a URL retrieved from web browser and search engine crawler perspectives. This work presents three methods of using difference in tags to determine whether a URL is cloaked. Since the tags of a web page generally do not change as frequently and significantly as the terms and links of the web page, tag-based cloaking detection methods can work more effectively than the term- or link-based methods. The proposed methods are tested with a dataset of URLs covering short-, medium- and long-term users' interest. Experimental results indicate that the tag-based methods outperform term- or link-based methods in both precision and recall. Moreover, a Weka J4.8 classifier using a combination of term and tag features yields an accuracy rate of 90.48%.

1. Introduction

Spam email has increased in severity over the past decade, with anti-spam legislation (CAN-SPAM, 2003) and technologies (Fdez-Riverola, Iglesias, Díaz, Méndez, & Corchado, 2007; Gordillo & Conde E., 2007; Hsiao & Chang, 2008) receiving considerable attention. Meanwhile, web spam has emerged. Web spam attempts to influence search engine ranking algorithm in order to boost the rankings of specific web pages in search engine results. Since most internet users do not look beyond the first two pages of search engine results, boosting the ranking of a web page in the search engine results can generally translate into more visitors to the web page and consequently, more business. Search engines adopt sophisticated algorithms to rank web pages to provide the "best" results first in response to users' queries. However, if web spam becomes prevalent and no proper measures are taken by search engine companies, then the search results fail to reflect users' queries accurately, forcing users to traverse more pages to find their desired information. Hence, detecting web spam has been become very important for search engine companies provide high quality and competitive searching services.

To avoid detection, web spam generally adopts hiding techniques to conceal its real intention (Gyöngyi & Garcia-Molina, 2005). Cloaking is one such technique that is widely used for web spam. It means that the content presented to the search engines' crawlers is different from that presented to users' web browsers. For instance, when requesting a cloaked URL, the content received by a crawler could be a keyword-enriched web page, or a web page containing many anchor texts linking to other URLs. In either case, this content is not intended for human viewing, but only for boosting the ranking of specific URLs in the search engine results. However, the content received by a web browser could be a normal-looking, human-readable web page, or simply a broken link such as "HTTP 404: file not found" or "HTTP 503: service not available". Since regular users do not see the spammed content (i.e., the content received by crawlers), they are unaware that the URL is cloaked. Wu and Davison (2005) experimented with a set of popular queries and found that more than 8% of top 200 URLs returned from Google employed cloaking. Off-the-shelf cloaking software also abets the expansion of cloaking. Hence, search engine companies need to detect cloaked URLs in order to combat web spam.

Owing to the dynamic nature of web pages, cloaking cannot be detected accurately by simply checking whether the contents of a URL received by a browser and by a crawler are identical. This simple method falsely identifies many dynamically generated pages as cloaked because their content could be different on each access. Hence, most cloaking detection methods (Chellapilla & Chickering, 2006; Wu & Davison, 2005, 2006) request a URL more than twice using different identities (i.e., either as a browser or as a crawler), and then compare the differences among the received contents to discern whether the difference is caused by cloaking or simply by the dynamic nature of the URL. For instance, suppose that some cloaking detection method requests a URL twice from a crawler perspective, and once from a browser perspective. Three copies

* Tel.: +886 3 4638800/2611; fax: +886 3 4352077.
E-mail address: jun@saturn.yzu.edu.tw

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.09.056

of the URL then become available for comparison. Let C_1 and C_2 represent the two copies received from a crawler perspective, and B_1 represent the copy received from a browser perspective. Intuitively, if the difference between C_1 and B_1 is much greater than that between C_1 and C_2, then this URL is probably cloaked.

Most cloaking detection methods measure the difference between two copies of a URL according to the terms or links that they contain. These methods may falsely identify legitimate dynamically generated pages and frequently updated pages as cloaked, because such pages generally contain different terms and links on each access. Despite their variant nature, the structures of such pages generally do not change much within a short period of time (e.g., one day). Based on this observation, this study proposes tag-based methods for cloaking detection, where the tags in a web page are utilized to capture the structure of the web page. The tag-based methods are tested with a dataset of URLs covering short-, medium- and long-term users' interest. The results indicate that the tag-based methods outperform previous methods (Chellapilla & Chickering, 2006; Wu & Davison, 2005) in both precision and recall.

The contribution of this work is as follows. First, the tag-based methods are proposed for cloaking detection, and their performance is compared against the term- and link-based methods. Second, a taxonomy is presented to categorize various cloaking detection methods. Third, a new type of cloaking (called dynamic cloaking) is described and discussed. The rest of this paper is organized as follows. Section 2 reviews related work on cloaking detection. Section 3 introduces the tag-based cloaking detection methods. Section 4 then presents the taxonomy of cloaking detection methods. Section 5 describes the dataset adopted in the experiment, and Section 6 presents the performance results. Section 7 describes dynamic cloaking. Conclusions are finally drawn in Section 8.

2. Related work

This section describes cloaking methods, and surveys pertinent literature on cloaking detection. An overview of web spam and its detection can be found in Gyöngyi and Garcia-Molina (2005).

2.1. Cloaking methods

A web site adopting cloaking must discern whether an HTTP request is issued by a search engine's crawler or by a user's web browser. This distinction is normally based on the following data:

- IP address: The cloaking web site maintains a list of the IP addresses of the major search engines' crawlers. If the IP address of the client making the HTTP request is on the list, then this client is deemed a crawler, and otherwise a web browser. This type of cloaking is called *IP cloaking*. Since the IP addresses are difficult to fake, a cloaking web site can recognize its visitor's identity quite accurately if the list of crawlers' IP addresses is up-to-date. Many cloaking software packages support IP cloaking, and some even provide services to periodically refresh the list of crawlers' IP addresses.
- *User-agent* HTTP header: The *user-agent* header specifies the application originating the HTTP request. Because browser and crawler applications generally have different user-agent headers, they can be distinguished accordingly. However, *user-agent* header can easily be faked. For instance, a crawler might try to hide its true identity by placing the *user-agent* of some browser application into its HTTP requests.
- *Referer* HTTP header: The *referer* header specifies the IP address of the referrer of the HTTP request. A cloaking web site can utilize this information to discern between a crawler and a browser, since a regular surfer usually visits the site from a search engine or another website, while a crawler generally does not have any referrer (Edelman, 2004). However, the *referer* header, like the *user-agent* header, can easily be faked.

Based on the above data, a web site that adopts cloaking can present different contents to browsers and crawlers. This is generally achieved by configuring the web server accordingly (e.g., specifying URL *rewriting rules* in Apache HTTP Server Module mod_rewrite, 2008) or by server-side scripting.

2.2. Cloaking detection methods

Most cloaking detection methods compare multiple copies of a URL, where some copies are retrieved from a crawler's perspective, and some from a web browser's perspective. For brevity, the rest of this study uses B_i and C_i to denote the contents of a URL retrieved by a browser and a crawler for the ith time, respectively.

Najork (2005) developed a patented method for detecting cloaked pages. His method classifies an URL as clocked if the *representations* of B_1 and C_1 of the URL do not match. This method works efficiently, because it retrieves a URL only twice. However, for the same reason, this method is likely to falsely classify dynamically generated or frequently updated pages as cloaked.

Wu and Davison (2005) proposed three methods to detect cloaking. Their methods treat B_i and C_i of a URL either as a set of terms or as a set of links. Their first method, called TermDiff3, counts the number of different terms between C_1 and C_2 (denoted as TC_1C_2) and the number of different terms between B_1 and C_1 (denoted as TB_1C_1). Notably, each distinct term is only counted once. If $TB_1C_1 > TC_1C_2$ by a predefined threshold, then the URL is classified as cloaked. Similarly, their second method, called LinkDiff3, counts the number of different links between C_1 and C_2 (denoted as LC_1C_2) and the number of different links between B_1 and C_1 (denoted as LB_1C_1), and if $LB_1C_1 > LC_1C_2$ by a predefined threshold, then this URL is classified as cloaked. Both TermDiff3 and LinkDiff3 need to retrieve a URL three times (i.e. B_1, C_1 and C_2). They both do not yield satisfactory results, since a high (or low) threshold improves precision (or recall) at the expense of very low recall (or precision). Their third method, called TermDiff4, counts the number of terms occurring in both B_1 and B_2 but not in C_1 or C_2 (denoted as TBNC), and the number of terms occurring in both C_1 and C_2 but not in B_1 or B_2 (denoted as TCNB). If TBNC + TCNB is greater than a predefined threshold, then this URL is classified as cloaked. Although TermDiff4 needs to retrieve a URL four times (i.e. B_1, B_2, C_1 and C_2), it performs much better than either TermDiff3 or LinkDiff3, in terms of both precision and recall.

Chellapilla and Chickering (2006) proposed using the normalized term frequency difference (NTFD) to assess the difference between two copies of a URL. In contrast to Wu and Davison's methods, this method treats each copy of a URL as a multiset of terms, and extends the set operations for multisets in order to calculate NTFD. Let T_1 and T_2 denote any two multisets. The term frequency difference (TFD) between T_1 and T_2 is thus calculated as follows:

$$TFD(T_1, T_2) = |\ (T_1 \setminus T_2) \cup (T_2 \setminus T_1)\ | = |\ T_1 \cup T_2\ | - \mathbf{2}\ |\ T_1 \cap T_2\ |$$

where $|T|$ represents the size of a multiset T. Notably, unlike normal set operations, the weighting of $|T_1 \cap T_2|$ is doubled in the above equation, and $|T_1 \cup T_2|$ is calculated from $|T_1| + |T_2|$. The value of NTFD(T_1, T_2) is then calculated by dividing TFD(T_1, T_2) by $|T_1 \cup T_2|$ to normalize the value to within the interval [0,1]. Chellapilla and Chickering argued that NTFD could reduce the bias against large web pages, but at the cost of raising the bias against small web pages. Therefore, the benefit of NTFD over TFD requires further consideration. Based on NTFD, the cloaking score, S, of a URL is given by $\frac{\Delta_D}{\Delta_S}$, where Δ_D = min{NTFD(C_1, B_1), NTFD(C_2, B_1)}, and Δ_S = max{NTFD

(C_1, C_2), NTFD(B_1, B_2)}. If S is above a predefined threshold, then this URL is classified as cloaked. Choosing an appropriate threshold for S is not very intuitive, since S could be any real number within the interval $[0, \infty)$. Their experimental results indicate that this method performs quite well on the set of URLs retrieved from monetizable queries (i.e., queries whose search engine results contain sponsors' links).

Wu and Davison (2006) presented a two-step cloaking detection method. In the first step, their method retrieves a set of candidate URLs, where TB_1C_1 (and/or LB_1C_1) of each candidate URL is above a predefined threshold. Since most URLs return identical content to a browser and a crawler (about 74%, according to Chellapilla and Chickering (2006)), this step usually filters out most non-cloaked URLs. In the second step, a set of features is generated from B_1, B_2, C_1 and C_2 of each candidate URL. These features include content-based features from each copy (such as the response code, the number of terms and whether a specific attribute exists), link-based features from each copy (such as the number of total links, the number of relative links and the ratio of number of relative links to the number of total links), and features for two corresponding copies (such as $|B_1 \cap C_1|$, $|B_1 \backslash C_1|$, and whether $|B_1| = |B_2|$). Finally, a C4.5 decision tree based on these features is constructed to identify cloaking.

3. Tag-based cloaking detection

The intuition behind tag-based cloaking detection methods is that the content of a legitimate dynamically generated webpage generally employs a similar (or even the same) structure on each access. In contrast, the structure of the content of a cloaked URL presented to a crawler is generally different from that presented to a browser. A cloaked URL can thus be distinguished from a non-cloaked URL.

Tag-based methods use the tags in a webpage to represent its structure. More specifically, tag-based methods consider every copy of a URL as a *multiset* or a *sequence* of tags. A webpage usually has a very small variety of tags. Therefore, the tag-based method considering a copy of a URL as a *set* of tags does not capture the properties of the URL effectively, and thus is not adopted.

Three tag-based cloaking detection methods (TagDiff2, TagDiff3 and TagDiff4) are presented below. They all consider every copy of a URL as a *multiset*, and extend set operations (union, intersection and difference) to multisets by viewing a multiset as a sequence and treating each occurrence of any element x in a multiset as different, and the ith occurrences of each x in two multisets as identical. Standard set operations are then directly applied on multisets. For instance, given two multisets $T_1 = \{a,a,b,b,b,d,d\}$ and $T_2 = \{a,c,b,a,a,b\}$, we have $T_1 \cap T_2 = \{a,a,b,b\}$, $T_1 \backslash T_2 = T_1 \backslash (T_1 \cap T_2) = \{b,d,d\}$, and $T_1 \cup T_2 = \{a,a,a,b,b,b,c,d,d\}$. Moreover, for two multisets T_1 and T_2, we have $|(T_1 \backslash T_2) \cup (T_2 \backslash T_1)| = |(T_1 \cup T_2)| - |(T_1 \cap T_2)|$, which is similar to the standard set operation but different from the multiset operation defined by Chellapilla and Chickering (2006).

Each of these three methods defines its own formula to calculate the difference between the browser's copies and the crawler's copies of a URL. If this difference exceeds a predefined threshold, then the URL is classified as cloaked. The formulae for the three methods are shown below:

- TagDiff2: $| B'_1 \setminus C'_1 | + | C'_1 \setminus B'_1 |$
- TagDiff3: $(| B'_1 \setminus C'_1 | + | C'_1 \setminus B'_1 |) - (| C'_1 \setminus C'_2 | + | C'_2 \setminus C'_1 |)$
- TagDiff4: $| (B'_1 \cap B'_2) \setminus (C'_1 \cup C'_2) | + | (C'_1 \cap C'_2) \setminus (B'_1 \cup B'_2) |$

where B'_i and C'_i respectively denote the multisets of tags collecting from the copies B_i and C_i of a URL. These formulae clearly reveal that TagDiff2, TagDiff3 and TagDiff4 have to retrieve a URL two, three and four times, respectively. Additionally, the formulae for TagDiff3 and TagDiff4 are similar to those of TermDiff3 and TermDiff4, respectively. Section 6 compares the performance of these three methods against the term-based and link-based methods.

4. Taxonomy of cloaking detection methods

This section presents a taxonomy of cloaking detection methods. This taxonomy can be used to derive new cloaking detection methods. It categorizes cloaking detection methods along four dimensions, copy, element, group and formula.

The "*copy*" dimension specifies the maximum number of copies of a URL needed by a cloaking detection method. Some methods (e.g., Chellapilla and Chickering, 2006; Wu and Davison, 2006) first retrieve B_1 and C_1 of a URL, then retrieve B_2 and C_2 only when B_1 and C_1 are insufficient for them to discern whether the URL is cloaked. The copy dimension for such methods is then regarded as 4. Since accurate cloaking detection is unlikely from retrieving a URL only once, the smallest value for this dimension is 2. Common values for this dimension are between 2 and 4. A value greater than 4 is also possible, but may incur too much overhead, while adding little to cloaking detection.

The "*element*" dimension specifies which elements from a web page are used for cloaking detection. The possible values along this dimension are terms, links and tags, since an HTML web page comprises these three types of elements. These elements can also be used in combination by a cloaking detection method. For instance, Section 6 describes the construction of a Weka J4.8 classifier using both term and tag elements for cloaking detection.

The "*group*" dimension specifies how data elements are treated as a group. Some possible values for "*group*" are "set", "multiset" and "sequence". For instance, a cloaking detection method may consider a web page as a set of links, a multiset of tags or a sequence of terms. The value of this dimension can also influence how computation is performed in the next dimension, namely formula. The operations on sets are well-defined, while the operations on multisets and sequences are not. For instance, both CloakingScore (Chellapilla and Chickering, 2006) and TagDiff4 are based on multisets, but they define operations on multisets differently, as discussed in Section 3.

The "edit distance" (Gusfield, 1997) might be adopted to quantify the difference between two sequences. However, the edit distance can be calculated in many different ways, in terms of the weights assigned to "insert", "delete" and "update" operations. Although sequences preserve more information than sets and multisets, calculating the distance between two sequences may be too costly for identifying cloaking.

The "*formula*" dimension defines the formula used to measure the difference between the browser's copies and the crawler's copies. For instance, in TermDiff4, the difference is calculated as TBNC + TCNB, while in CloakingScore, it is calculated as $\frac{\Delta_D}{\Delta_S}$. Additionally, some methods (Wu and Davison, 2006) simply derive the difference using machine learning algorithms, and classify URLs as cloaked or non-cloaked accordingly. Table 1 summarizes all cloaking detection methods along these four dimensions.

5. Data sets

Several sets of queries were collected from the Web. Those queries were then entered into Google to retrieve several sets of URLs. URLs were selected randomly from those sets of URLs. These selected URLs were then used to evaluate the performance of various cloaking detection methods, as described in Section 6. The data collection process is described below.

Table 1
Taxonomy of cloaking detection methods

Methods	Copy	Element	Group	Formula
Najork (Najork, 2005)	2	Terms + links + tags	Sequence	$\|Rep(B_1) - Rep(C_1)\|$
TermDiff3 (Wu & Davison, 2005)	3	Terms	Set	$TB_1C_1 - TC_1C_2$
LinkDiff3 (Wu & Davison, 2005)	3	Links	Set	$LB_1C_1 - LC_1C_2$
TermDiff4 (Wu & Davison, 2005)	4	Terms	Set	TBNC + TCNB
CloakingScore (Chellapilla & Chickering, 2006)	4	Terms	Multiset	$\frac{\Delta_D}{\Delta_S}$
Wu & Davison (Wu & Davison, 2006)	4	Terms + links + tags	Set + multiset	C4.5 Algorithm
TagDiff2	2	Tags	Multiset	$\| B'_1 \setminus C'_1 \| + \| C'_1 \setminus B'_1 \|$
TagDiff3	3	Tags	Multiset	$\| (B'_1 \setminus C'_1 \| + \| C'_1 \setminus B'_1 \|) - (\| C'_1 \setminus C'_2 \| + \| C'_2 \setminus C'_1 \|)$
TagDiff4	4	Tags	Multiset	$\| (B'_1 \cap B'_2) \setminus (C'_1 \cup C'_2) \| + \| (C'_1 \cap C'_2) \setminus (B'_1 \cup B'_2) \|$

5.1. Query data sets

A spammed webpage is often stuffed with terms that users are likely to find interesting. Therefore, this performance study began by collecting four sets of queries from the Web, where each set represented short, medium or long-term of users' interest. The first set (denoted as WQ) contained Google's *weekly* popular queries[1] from 2006/1/1 to 2006/12/23. This set comprised 710 queries, reduced to 595 after removing duplicates. The second set (denoted as MQ) contained monthly popular queries in year 2006 from both Google[1] and AOL.[2] This set comprised 775 queries (665 from Google and 110 from AOL), reduced to 484 after removing duplicates. The third set (denoted as YQ) contained the *yearly* popular queries from six sources in year 2006: Google,[3] Yahoo,[4] MSN,[5] AOL,[6] Ask.com[7] and Lycos.[8] The numbers of queries from these six sources were respectively 20, 100, 70, 210, 50 and 160. This set contained 402 queries after removing duplicates. The fourth set (denoted as DQ) comprised the directory names from the top two levels of ODP,[9] but excluding the descendants of the "World" directory to avoid adding 79 language names to DQ. DQ had 599 directory names, reduced to 483 after removing duplicates.

Intuitively, WQ, MQ and YQ respectively reflect the short, medium and long-term popularity of terms to web users. In contrast, DQ represents an even longer term of popularity, since the directory names at the top two levels of a classified catalog (e.g., ODP) for WWW change infrequently. Notably, many queries appear in more than one query data set.

5.2. URL data sets

Every query in these four sets of queries was entered into Google, and its top 200 search results were collected. The search results for queries in the same query set were stored together to form a set of URLs. Hence, four sets of URLs (denoted as WU, MU, YU and DU) were respectively built from the four sets of queries (i.e., WQ, MQ, YQ and DQ).

The content of each URL in WU, MU, YU and DU was retrieved several times in the order below.

$$C_1 \rightarrow B_1 \rightarrow C_2 \rightarrow B_2$$

Here, C_i and B_i represent the ith time of retrieving this URL from the crawler and browser perspectives, respectively. Notably, C_1, B_1, C_2 and B_2 were retrieved consecutively.

URLs returning different HTTP codes to crawlers and browsers were excluded, since they are easily-detected examples of cloaked pages. This study focused only on the URLs that returned HTTP code 200 (meaning a successful request) for C_1, B_1, C_2 and B_2 (the 3rd column of Table 2), since cloaking is difficult to detect in such URLs. Therefore, these URLs suited for meaningfully comparing cloaking detection methods. URLs with $C_1 = B_1$ (the 4th column of Table 2) were also excluded, since they are easily identified as non-cloaked. Here, $C_1 = B_1$ means that the raw content of C_1 and B_1 are identical. Finally, we randomly sampled 1% of URLs from the remaining URLs (see the 5th column of Table 2 and the 2nd column of Table 3) in each URL data set, and then manually checked whether each URL in the sample was cloaked (see the 3rd column of Table 3).

All sampled URLs were put together to form a URL data set containing 1344 URLs with 394 (29.32%) cloaked URLs. This URL data set was used to evaluate the performance of various cloaking detection methods, as described next.

6. Cloaking detection results

This section compares the performance of the proposed tag-based cloaking detection methods against term-based and link-based methods. All of these methods require setting a threshold value no less than zero. Hence, the performance of a method is evaluated by measuring the precision and recall at different threshold values. In this study, the precision is calculated as the percentage of cloaked URLs in the URLs predicted to be cloaked, and the recall is calculated as the percentage of cloaked URLs in the data set that are correctly identified.

Table 4 shows the precision and recall of the three tag-based methods. TagDiff2, TagDiff3 and TagDiff4 all achieved their maximal recall at a threshold value of 0. TagDiff4 had a smaller maximum recall than the other two methods, revealing that TagDiff4 is more cautious in predicting a URL as cloaked. However, TagDiff4 still maintains a very good precision of 85.22% at its maximum recall. Conversely, TagDiff2 is more aggressive at predicting a URL as cloaked, but had a precision of only 72.11% at its maximum recall.

Table 5 shows the precision and recall values of two term-based methods TermDiff4 and CloakingScore. Other term-based methods (i.e., TermDiff2 and TermDiff3), are not presented here because they are inferior to TermDiff4 and CloakingScore. The results reveal that TermDiff4 had difficulty in raising the recall value (similar to TagDiff4), while CloakingScore suffered from poor precision. Table 6 shows that LinkDiff3 suffered from both poor precision and poor recall.

Notably, the threshold has a different meaning in each method. Therefore, the best method cannot be identified simply by directly

[1] Weekly Lists of 2006 Zeitgeist Archive at http://www.google.com/press/zeitgeist/archive2006.html.
[2] AOL's Monthly Top Searches from http://about-search.aol.com/hotsearches2006/[january-november].html.
[3] 2006 Year-End Google Zeitgeist at http://www.google.com/press/zeitgeist2006.html.
[4] Yahoo's Top Searches of 2006 from http://buzz.yahoo.com/topsearches2006/lists/.
[5] MSN's Top of Live Search for 2006 from http://blogs.msdn.com/livesearch/archive/2006/12/14/the-top-of-live-search-for-2006.aspx.
[6] AOL's Year's Top Searches from at http://about-search.aol.com/hotsearches2006/index.html.
[7] Ask.com's Top Searches from http://websearch.about.com/od/topsearches2006/a/topsearchesask.htm.
[8] Lycos 50 Archives from http://50.lycos.com/archives.asp.
[9] ODP at http://dmoz.org/.

Table 2
URL data sets

URL set	# of URLs	# of URLs with HTTP = 200	# (%) of URLs with HTTP = 200 and $C_1 = B_1$	# (%) of URLs with HTTP = 200 and $C_1 \neq B_1$
WU	117025	105612	59222 (56.08%)	**46390** (**43.92**%)
MU	91842	82712	48253 (58.34%)	**34459** (**41.66**%)
YU	77667	69573	38431 (55.24%)	**31142** (**44.76**%)
DU	93600	83031	60792 (73.22%)	**22239** (**26.78**%)

Table 3
Sample of URL data sets

URL set	# of sampled URLs	# (%) of cloaked URLs
WU	464	128 (27.59%)
MU	345	93 (26.96%)
YU	312	101 (32.37%)
DU	223	72 (32.29%)
Total	1344	394 (29.32%)

Table 4
Precision and recall for tag-based methods

Threshold	TagDiff2		TagDiff3		TagDiff4	
	Precision	Recall	Precision	Recall	Precision	Recall
0	0.7211	0.9645	0.8350	0.8477	0.8522	0.8198
1	0.7986	0.8655	0.8684	0.7538	0.8864	0.7132
2	0.8303	0.8071	0.8850	0.7030	0.9164	0.6675
3	0.8434	0.7792	0.8986	0.6751	0.9247	0.6548
4	0.8642	0.7107	0.9170	0.6168	0.9325	0.5964
5	0.8742	0.6701	0.9173	0.5914	0.9417	0.5736
6	0.8877	0.6421	0.9300	0.5736	0.9476	0.5508
7	0.8914	0.6041	0.9280	0.5558	0.9457	0.5305
8	0.8915	0.5838	0.9345	0.5431	0.9488	0.5178
9	0.8916	0.5635	0.9358	0.5178	0.9510	0.4924
10	0.9083	0.5533	0.9479	0.5076	0.9598	0.4848
20	0.9409	0.4848	0.9626	0.4569	0.9667	0.4416
40	0.9686	0.3909	0.9728	0.3629	0.9790	0.3553
320	1.0000	0.1091	1.0000	0.1015	1.0000	0.0964

Table 5
Precision and recall for term-based methods

CloakScore			TermDiff4		
Threshold	Precision	Recall	Threshold	Precision	Recall
0	0.360835	0.92132	0	0.622407	0.761421
0.02	0.362086	0.916244	1	0.672249	0.713198
0.11	0.357732	0.880711	2	0.740331	0.680203
0.27	0.361111	0.857868	4	0.792453	0.639594
0.53	0.371293	0.794416	8	0.857664	0.596447
0.78	0.395918	0.738579	16	0.901288	0.532995
0.94	0.439441	0.718274	32	0.952941	0.411168
1.1	0.805556	0.662437	64	0.972222	0.266497
1.98	0.81685	0.56599	128	1	0.170051
19.93	0.859155	0.309645	256	1	0.142132

comparing the performance of two methods at the same threshold value. Fig. 1 shows the recall-precision curves, which better compare the performance of these methods. The curve for TagDiff3, which was omitted to avoid crowding Fig. 1, lies very close to the curve for TagDiff4. The curves for the tag-based methods lie above that of the other methods, revealing that the tag-based methods are more effective than the other methods. The curve for TagDiff4 exhibited the best performance. The precision of TagDiff2 was smaller than that of TagDiff4, while the maximal recall of TagDiff2 was 96.45%, which is much greater than that of TagDiff4. Moreover, TagDiff2 only needs to retrieve a URL twice, making it efficient.

To evaluate the combined effect of using both term- and tag-based features, a Weka J4.8 classifier (Witten and Frank, 2005) was constructed with TagDiff2, TagDiff3, TagDiff4, TermDiff3 and TermDiff4 as input. The classifier yielded an accuracy rate 90.48% on our sampled URL dataset using 10-fold cross-validation.

Table 6
Precision and recall for LinkDiff3

Threshold	Precision	Recall
0	0.711246	0.593909
1	0.713311	0.530457
5	0.781395	0.426396
10	0.81768	0.375635
20	0.824324	0.309645
35	0.827068	0.279188
55	0.814516	0.256345
80	0.8125	0.230964
110	0.86	0.218274
145	0.890244	0.185279
185	0.919355	0.14467

7. Dynamic cloaking

All cloaking detection methods discussed so far assume that the content of a cloaked URL sent to a crawler is *always* different from that sent to a web browser. This cloaking behavior is herein called "*static cloaking*". This section introduces another type of cloaking, called "*dynamic cloaking*".

A cloaker (i.e., a cloaked URL) is analogous to a villain in our society. A static cloaker is like a villain who constantly engages in wrongdoing, and thus is more likely to get caught. In contrast, a dynamic cloaker is like a villain who sometimes behaves like a decent person, and making it difficult to identify. A clever villain certainly behaves himself when he is aware of police presence nearby. *Dynamic cloaking* is the practice of cloaking intermittently to make it difficult to identify. Several URLs from the URL datasets described in Section 5 were found to be possible cases of dynamic cloaking. Some of these met the conditions $C_1 = B_1$ and $C_2 \neq B_2$, and were classified as non-cloaked by all cloaking detection methods, while others met the conditions $C_1 \neq B_1$ and $C_2 = B_2$.

To combat dynamic cloaking, this study first explores possible dynamic cloaking behaviors. A dynamic cloaker can switch between "Cloak" and "NoCloak" modes. During "Cloak" mode, the cloaker behaves as a static cloaker, i.e., returns different content to crawlers and web browsers. Here, the content received by a crawler is said to be the "spammed version", since it is used to manipulate search engines' ranking algorithms, while the content received by a web browser is called the "non-spammed version". The cloaker always returns identical content (i.e., the "non-spammed version) to crawlers and web browsers during "NoCloak" mode. Switching between these two modes can be event-driven, as described below.

- The cloaker initially operates in "Cloak" mode.
- A "crawler event" is the event where the cloaker receives a request from a crawler. When a "crawler event" occurs, a timer recording the time of the last crawler request is reset. If the cloaker is in "Cloak" mode, then it sends the spammed version of the URL to the crawler, and switches itself to "NoCloak" mode.

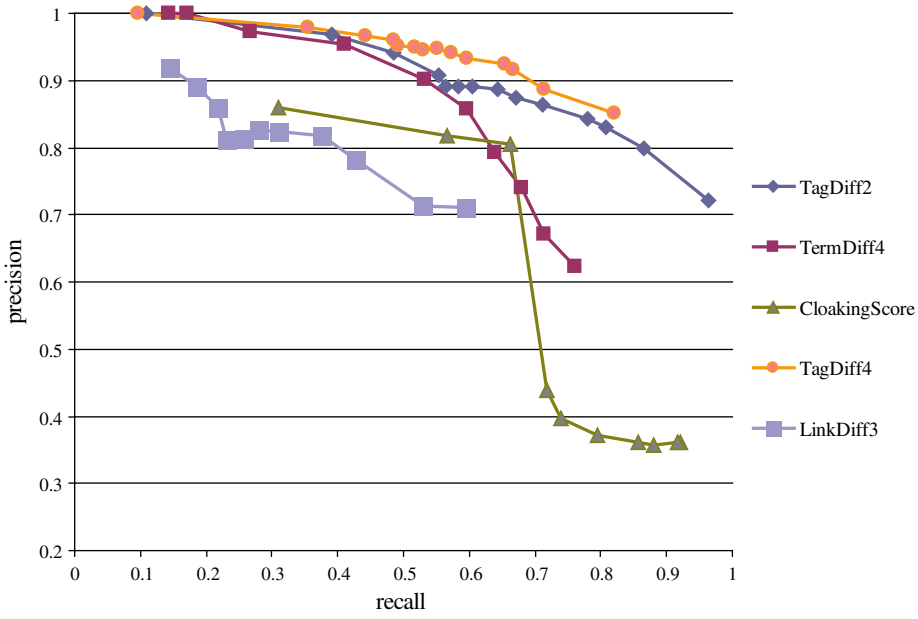

Fig. 1. Recall-precision curves of various cloaking detection methods.

- A "time up event" is the event that the time from the last crawler request exceeds a predefined threshold. The cloaker switches itself back to "Cloak" mode whenever this event occurs.

Event-driven dynamic cloaking is analogous to a villain (i.e., the cloaker) who behaves himself (i.e., stays in "NoCloak" mode) when he is aware of police presence nearby (i.e., a recent "crawler event"). However, if the police are far away (i.e., a "time up event"), then the villain is back to his evil-doing (i.e., switching back to "Cloak" mode).

Instead of maintaining a timer for all crawlers as described above, event-driven dynamic cloaking can also work at a finer granular level by storing the time of the most recent visit from each crawler. This can be achieved by maintaining a table of entries, each containing the IP address of a crawler and the time of its most recent request. This table, called the "monitor table", is initially empty, and is subsequently updated according to the following three events, corresponding to insertion, update and deletion of an entry in the monitor table.

- A "new crawler event" is the event that the cloaker receives a request from a crawler whose IP address is not in the monitor table. The cloaker replies to a "new crawler event" with the spammed version of the URL to the crawler, and adds the IP of the crawler and the current time to the monitor table.
- An "old crawler event" is the event that the cloaker receives a request from a crawler whose IP address is already in the monitor table. The cloaker replies to an "old crawler event" with the non-spammed version of the URL to the crawler, and updates the time of the entry of this crawler in the monitor table to the current time.
- A "time up event" is the event that the times of some entries are earlier than the current time by a predefined threshold. This event causes all such entries to be removed from the monitor table.

Event-driven dynamic cloaking may produce a URL satisfying $B_1 = B_2 = C_2 \neq C_1$. TermDiff4 would incorrectly classify this URL as cloaked, since TBNC = TCNB = 0. Similarly, CloakingScore would also incorrectly classify this URL as cloaked, since $\Delta_D = 0$ and therefore the cloaking score $S = 0$. Such cloaking is difficult to identify using current methods.

Given a URL satisfying $B_1 = B_2 = C_2 \neq C_1$ in this event-driven dynamic cloaking scenario, only C_1 is the spammed version. Hence, rather than attempt to identify dynamic cloaking, search engines can avoid the impact of spamming by simply using the crawler's copy that is not spammed (i.e., C_2, the copy identical or similar to the browser's copies) for their ranking algorithms.

8. Conclusion

This work proposes tag-based methods for cloaking detection. Experimental results reveal that tag-based methods are more effective than both term and link-based methods. Moreover, TagDiff2 performs quite well, even though it only requests a URL twice. TagDiff2 yields a very high recall at a low threshold value. As described in Section 5.2, URLs with $C_1 = B_1$ are excluded from our datasets because they are considered as non-cloaked URLs. Experimental results reveal that this step can be replaced by using TagDiff2 with a low threshold (e.g., 0) to exclude additional URLs, thus reducing the number of URLs that must be retrieved more than twice. TagDiff4 can also be adopted to increase the precision of the detection result. Alternatively, a classifier that combines term- and tag-based features can also detect cloaking very accurately.

This work presents a taxonomy of cloaking detection methods. This taxonomy can be utilized to systematically compare different methods, and develop new cloaking detection methods. This work also introduces dynamic cloaking, which is harder to identify than conventional static cloaking. Careful examination of the behavior of dynamic cloaking reveals that handling dynamic cloaking involves not only appropriate cloaking detection methods, but also selecting the correct copy to be used in search engine ranking algorithms.

References

Apache Module mod_rewrite (2008), http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html.

CAN-SPAM Act of 2003, http://www.legalarchiver.org/cs.htm.

Chellapilla, K., & Chickering, D. M. (2006). Improving cloaking detection using search query popularity and monetizability. In *Proceedings of the second international workshop on adversarial information retrieval on the web*. Seattle, USA.

Edelman, B. (2004) WhenU Spams Google, Breaks Google "No Cloaking" Rules. http://www.benedelman.org/spyware/whenu-spam/.

Fdez-Riverola, F., Iglesias, E. L., Díaz, F., Méndez, J. R., & Corchado, J. M. (2007). Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications, 33*(1), 36–48.

Gordillo, J., & Conde, E. (2007). An HMM for detecting spam mail. *Expert Systems with Applications, 33*(3), 667–682.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences: Computer science and computational biology*. Cambridge University Press.

Gyöngyi, Z., & Garcia-Molina, H. (2005). Web spam taxonomy. In *Proceedings of the first international workshop on adversarial information retrieval on the web*. Chiba, Japan.

Hsiao, W.-F., & Chang, T.-M. (2008). An incremental cluster-based approach to spam filtering. *Expert Systems with Applications, 34*(3), 1599–1608.

Najork, M. A. (2005). System and method for identifying cloaked web servers. United States Patent 6910077 Issued on June 21, 2005.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* (2nd ed.). Morgan Kaufmann.

Wu, B., & Davison, B. D. (2005). Cloaking and redirection: A preliminary study. In *Proceedings of the first international workshop on adversarial information retrieval on the web*. Chiba, Japan.

Wu, B., & Davison, B. D. (2006). Detecting semantic cloaking on the web. In *Proceedings of the 15th international conference on world wide web* (pp. 819–828). Edinburgh, Scotland.