Detecting Cloaking Web Spam Using Hash Function

Shekoofeh Ghiam^{1,*}, Alireza Nemaney Pour²

¹Faculty of Computer Engineering, Sharif University of Technology, International Campus, Kish Island, Hormozgan, Iran

²Faculty of Computer Engineering, Abhar Islamic Azad University, Abhar, Iran

*Corresponding Author: ghiam@kish.sharif.edu

Copyright © 2013 Horizon Research Publishing All rights reserved.

Abstract Web spam is an attempt to boost the ranking of special pages in search engine results. Cloaking is a kind of spamming technique. Previous cloaking detection methods based on terms/links differences between crawler and browser's copies are not accurate enough. The latest technique is tag-based method. This method could find cloaked pages better than previous algorithms. However, addressing the content of web pages provides more accurate results. This paper proposes an algorithm, working based on term differences between crawler and browser's copies. In addition, dynamic cloaking, which is a new and complicated kind of cloaking, is addressed. In order to increase the speed of comparison, we introduce hash value, calculated by Hash Function. The proposed algorithm has been tested with a data set of URLs. Experimental results indicate that our algorithm outperforms previous methods in both precision and recall. We estimate that about 9% of all URLs in data set utilize static cloaking and about 2% of all URLs utilize dynamic cloaking.

Keywords Cloaking, Hash Function, Precision, Recall, Web Spam

1. Introduction

Suppose a scenario in which you search a query in a popular search engine like Google but do not find any relevant answer even from top results. This is called web spam, which is an attempt to manipulate search engine ranking algorithm in order to boost the ranking of special pages in search engine results. There are three different goals for uploading a spam page via spammers. The first is to attract viewers to visit their sites to enhance the score of the page in order to increase financial benefits for the site owners. The second goal is to encourage people to visit their sites in order to introduce their companies and the products, and to persuade visitors to buy their productions. The last goal, which is the worst case, is to install malware on victim's computer.

Spam pages may waste important resources of search engines such as wasting network bandwidth while crawling, wasting CPU cycles while processing, wasting storage space

while indexing, and wasting disk bandwidth while matching. Therefore, search engines use sophisticated algorithms to rank web pages in order to avoid giving high ranking to spam pages. However, spammers utilize different techniques to reach their goals. Cloaking is a kind of hiding technique that is widely used recently. In cloaking method, spammers try to deliver different contents to web crawlers and normal visitors in order to deceive search engine ranking algorithms. A web crawler is a computer program that browses the World Wide Web in a methodical, automated manner. A web crawler has a database, which stores the record of requested pages. Hence, search engine ranking algorithms use web crawlers to evaluate web pages, so spammers try to deliver deceptive version of a URL to web crawlers to get higher score. For example, when a cloaked URL is requested, the content delivered to the web crawler can be a page with lots of anchor text linking to other URLs or a page with lots of keywords. However, the page delivered to the web browser can be normal web page or a broken link such as "HTTP 404: file not find" or "HTTP 503: service not available".

According to Wu and Davison research [1], there are two kinds of cloaking; syntactic cloaking and semantic cloaking. Syntactic cloaking includes all situations in which different content is sent to a crawler versus real users. Semantic cloaking is a subset of syntactic cloaking and is an attempt to deliver different content to web crawler and web browser, so differences in meaning between these two copies of a URL may deceive search engine ranking algorithm. Experimental results of previous works in cloaking field show that web spam is growing rapidly and spammers try to boost their ranking on search engine results as much as possible, so detecting cloaked URLs is very essential for companies to combat web spam.

In this work we propose an algorithm working based on term differences between crawler and browser's copies. We checked URLs in different parts of the algorithm and tried to make it as accurate as possible. We addressed dynamic cloaking too.

The rest of this paper is as follows. Section 2 reviews some previously proposed cloaking detection methods and discuss the weakness of those methods. Section 3 explains our proposing algorithm. Section 4 is about creating our data set. Section 5 shows the evaluation results. Finally we conclude our discussion in section 6.

2. Related Works

Spammers use three different techniques for manipulating search engine ranking algorithms. These are: link-based techniques, content-based techniques and hiding techniques. There are some methods that have been proposed for detecting link spam [2-3-4-5-6-7]. Also there are some other methods for detecting content-based spam [8-9-10]. However, there are a few publications for hiding techniques and cloaking detection methods. The details can be found in [11].

According to Gyöngyi and Garcia-Molina research [12], cloaking is a kind of hiding technique. Spammers can identify search engine web crawlers by their network IP address. They keep a list of web crawlers IP addresses and update it frequently. Also they can identify crawlers by their user-agent names.

Najork [13] proposed a primary method for detecting cloaking web spam comparing crawler and browser's copies of a requested URL. Let assume C_i is the crawler's copy and B_i is the browser's copy. If crawler and browser perspective is completely the same, this page is not a cloaked page. Otherwise it is classified as a cloaked page. This method is likely to falsely classify dynamically generated or frequently updated pages as cloaked. To avoid this kind of mistake, further copies of a URL is needed.

Wu et al. [14] used two methods to detect cloaking: Term differences and Link differences. In term differences method the algorithm calculates the number of different terms between C_1 and C_2 , called NCC and then calculates the number of different terms between C_1 and B_1 , called NBC. By using a predefined threshold, if $NBC > NCC$ this page has a high probability to be a cloaked page. They proposed three algorithms and based on retrieving URLs they called them TermDiff2, TermDiff3 and TermDiff4. The best algorithm between these three algorithms that could find more spam pages is TermDiff4. In link difference method the algorithm calculates the number of different links between C_1 and C_2 , called LCC and then calculates the number of different links between C_1 and B_1 , called LBC. By using a predefined threshold, if $LBC > LCC$, there is a high probability that this page is a cloaked page. In addition, they proposed an algorithm, which automatically detects syntactic cloaking. In this algorithms four copies of a URL is needed, two from the crawlers' perspective and two from the browsers' perspective. The algorithm calculates the number of common terms in B_1 and B_2 but not exist in C_1 or C_2 , called TBNC. Also calculates the number of common terms in C_1 and C_2 but not exist in B_1 or B_2 , called TCNB. If $TBNC + TCNB$ is greater than the predefined threshold, this page is classified as a cloaked page. However, finding a suitable threshold is a critical job.

Chellapilla et al. [15], proposed an algorithm, which calculates cloaking score for each requested URL. This algorithm uses normalized term frequency difference to check differences between two copies of a URL, which are sent to web crawler and web browser. At first, it retrieves C_1

and B_1 . If they are from the same HTML, there is no cloaking. If not, pages will be changed to the text. If texts are equal, there is no cloaking. If not, term differences between C_1 and B_1 are calculated. If the result is zero, there is no cloaking. Otherwise, C_2 and B_2 are retrieved. Then cloaking score (S) is calculated according to following formulas. If S is greater than the predefined threshold, this page is classified as a cloaked page. Otherwise, it is a normal page.

$$S = \frac{\Delta D}{\Delta S}$$

$$\Delta S = \max(D(C_1, C_2), D(B_1, B_2)) \quad (1)$$

$$\Delta D = \min(D(C_1, B_1), D(C_2, B_2))$$

Wu et al. [1], proposed a two-step cloaking detection method, which detects semantic cloaking on the web. In the first step, C_1 and B_1 are compared. To avoid classifying dynamically generated or frequently updated pages as cloaking, a satisfactory threshold is used. It means that if the term difference between C_1 and B_1 is greater than the predefined threshold, this page will be sent to the next step of the algorithm. Otherwise, this page is a normal page. Therefore, the algorithm can find uncloaked pages only by retrieving two copies of a URL instead of four copies. Hence cost will be decreased. Then, the candidate pages will be sent to the second step of the algorithm. In the second step, called classification step, C_2 and B_2 are retrieved and a set of features is generated based on C_1 , C_2 , B_1 and B_2 . Finally, for identifying cloaking, a $C_{4.5}$ decision tree is generated.

Lin [16], proposed tag based method to detect cloaking. The motivation behind this method is that tags in a web page do not change as much as links and terms in a web page. Based on this concept, three tag-based cloaking detection methods are proposed. In the first one, two copies of a URL are retrieved that are C_1 and B_1 . In the second method, three copies of a URL are retrieved which are C_1 , B_1 and B_2 . In the third method, C_1 , B_1 , C_2 and B_2 are retrieved. Based on union, intersection and difference, three algorithms were proposed, called TagDiff2, TagDiff3 and TagDiff4 respectively.

3. Proposed Method

This paper proposes an algorithm, which compares web crawler and web browser's copies based on term differences. Although, tags in a web page do not change as much as terms and links, cloaking method is a kind of content hiding technique and spammers try to deliver different content to crawler and browser, so checking the content of web pages sent to web crawlers and web browsers will give more accurate results rather than only checking the tags of those web pages. As, term based methods which have been proposed in cloaking field so far are not accurate enough, our proposing algorithm tries to make the algorithm more accurate by new ideas. This kind of detection method is called static cloaking detection. In addition, our proposing algorithm addresses dynamic cloaking which is a new and

complicated kind of cloaking.

In order to compare crawler and browser's copies we introduce Hash Function. The motivation is to increase the speed of comparison when determining whether the copies of those pages are the same or not. Let f be a Hash Function such as MD5, then $f(B_i)$ and $f(C_i)$ are the hash values of the browser and crawler copies, respectively. When two web pages are the same, the hash values calculated by Hash Function are also the same. Therefore, by this property we can easily compare crawler and browser's copies by only comparing the hash value of them.

3.1. Detecting Static Cloaking

Static cloaking is a situation in which the crawler and browser's copies are different. For detecting static cloaking, the algorithm is working as follows:

Step 1: $f(C_1) = f(B_1)$

In the first step, only $f(C_1)$ and $f(B_1)$ are compared to find pages that are certainly uncloaked (normal pages). If the hash values of these two copies are different, $f(C_1) \neq f(B_1)$, C_2 and B_2 are retrieved and $f(C_2)$ and $f(B_2)$ are calculated, so cost will decrease because normal pages are detected only by retrieving two copies of a URL instead of three or four copies. Experimental results show that about 56% of all URLs in our data set belong to this category.

Next, suppose a scenario in which $f(C_1) \neq f(B_1)$ and $f(C_2) \neq f(B_2)$. It means that there is a high probability for cloaking. However, evidences are not enough. To make our algorithm more accurate, we check other hash values in following steps:

Step 2: $f(C_1) \neq f(B_1), f(C_2) \neq f(B_2), f(C_1) = f(C_2), f(B_1) = f(B_2)$

We call this part of the algorithm HashStatic1. Since the hash values of crawlers' copies are equal, and the hash values of browsers' copies are equal too, but crawler and browser's copies are different, there is a high potential for cloaking.

However, because of very dynamic nature of web pages, to avoid classifying normal pages as cloaked, we calculate the term differences between C_1 and B_1 , called T_{C1B1} , so this parameter helps us to find spam pages accurately because it means the differences between crawler and browser's copies which causes cloaking. If T_{C1B1} is less than the predefined threshold, it means that this difference is only because of dynamic nature of web pages and this URL is not cloaked. Experimental results show that about 2.2% of all URLs in our data set belong to this category. If T_{C1B1} is greater than the predefined threshold, it means that spammers try to deliver different version of a web page to crawler and browser, so this URL is a cloaked URL. Experimental results show that about 4.5% of all URLs in our data set belong to this category.

- Parameter for identifying cloaking in this step: T_{C1B1} , which means the term differences between crawler and browser's copies.

Step 3: $f(C_1) \neq f(B_1), f(C_2) \neq f(B_2), f(C_1) = f(C_2), f(B_1) \neq f(B_2)$

We call this part of the algorithm HashStatic2. The only

difference between this step and the previous one is that $f(B_1)$ and $f(B_2)$ are not equal.

In comparison with previous step, since $f(B_1) \neq f(B_2)$, the probability of cloaking is low because the possibility of existing dynamically generated or frequently updated pages is high. Experimental results prove this claim. To avoid false positive, we make our algorithm more accurate by calculating the term differences between C_1 and B_1 , called T_1 , between C_2 and B_2 , called T_2 . Next, we calculate $T_1 \cup T_2$, which means the summation of term differences between the first and the second copies of crawler and browser. We called it T_{1+2} . In order to avoid calculating common term differences between T_1 and T_2 twice, we calculate $T_1 \cap T_2$, which means the intersection between term differences of the first and the second copies of crawler and browser. We called it $T_{1,2}$. Then subtract T_{1+2} from $T_{1,2}$ and called it T_{total} , which means the total term differences between two copies of crawlers and browsers without their intersection. If T_{total} is greater than the predefined threshold, this URL is a cloaked URL because the total term difference is greater than threshold. Experimental results indicate that about 2.7% of URLs in our data set belong to this category. If T_{total} is less than the predefined threshold, there is no cloaking and the term difference is only because of the dynamic nature of web pages. Experimental Results show that about 1% of URLs belongs to this category.

- Parameter for identifying cloaking in this step: T_{Total} , which means the total term differences between two copies of crawlers and browsers without their intersection.

As we mentioned earlier, since B_1 and B_2 are different in this part, the probability of cloaking should be less than previous part. Experimental results prove it. In previous part about 4.5% of URLs are cloaked and in this part about 2.7% of URLs are cloaked.

Step 4: $f(C_1) \neq f(B_1), f(C_2) \neq f(B_2), f(C_1) \neq f(C_2), f(B_1) \neq f(B_2)$

Since $f(C_1)$ and $f(C_2)$, $f(B_1)$ and $f(B_2)$ are different, such pages are changing rapidly and we call them very dynamic pages. Previous methods in cloaking detection may put these kinds of pages in cloaked category. However, these pages only change rapidly and are not cloaked.

Experimental results show that about 25% of all URLs in our data set belong to this category. In addition, we checked about 5% of pages in this category manually to find the accuracy of our algorithm in this part. Results show that the accuracy of this part is about 98.99%.

Step 5: $f(C_1) \neq f(B_1), f(C_2) \neq f(B_2), f(C_1) \neq f(C_2), f(B_1) = f(B_2)$

The only difference between this step and the previous step is that $f(B_1)$ and $f(B_2)$ are the same. In comparison to step 2, since $f(C_1)$ and $f(C_2)$ are different, result prove that the probability of cloaking has been decreased.

In this part we use Wu and Davison research [7] to determine whether this difference between $f(C_1)$ and $f(C_2)$ and this equality between $f(B_1)$ and $f(B_2)$ cause cloaking or not. To do so, we calculate the term differences between C_1 and C_2 , called T_{C1C2} and then calculate the term differences between B_1 and C_1 , called T_{B1C1} . If T_{C1C2} is greater than T_{B1C1} , it means that the differences between crawlers' copies

are greater than the differences between crawler and browser's copies so there is no cloaking. Evaluation results show that about 0.9% of all URLs in our data set belong to this category. If T_{CIC2} is less than T_{BIC1} , it means that the term differences between crawler and browser's copies is greater than the term differences between crawlers' copies so this URL is a cloaked URL. Results show that about 1.5% of URLs in our data set belongs to this category.

- Parameters for identifying cloaking in this step: T_{CIC2} which means the term differences between two copies of crawler and T_{BIC1} which means the term differences between the first copy of crawler and the first copy of browser.

As we mentioned earlier, since C_1 and C_2 are different in this part, the probability of cloaking should be less than step 2. Experimental results prove it. In step 2 about 4.5% of URLs are cloaked and in this part about 1.5% of URLs are cloaked.

3.2. Detecting Dynamic Cloaking

According to Lin research [16], there is a new and complicated kind of cloaking, called dynamic cloaking. A dynamic cloaked URL behaves like a normal page making it difficult to be identified. The situation is described as $B_1 = B_2 = C_2 \neq C_1$. This paper proposes the following steps to detect such URLs by introducing Hash Function described above.

Step 6: $f(C_1) \neq f(B_1), f(C_2) = f(B_2), f(B_1) \neq f(B_2)$

Since $f(B_1)$ is not equal to $f(B_2)$, the situation of dynamic cloaking is not satisfied and there is no cloaking. Evaluation result shows that about 1.5% of all URLs in our data set belong to this category.

Step 7: $f(C_1) \neq f(B_1), f(C_2) = f(B_2), f(B_1) = f(B_2)$

We call this part of the algorithm HashDynamic. This is the situation for dynamic cloaking. It means that for the first time, when we retrieved the crawler's copy, it behaves like a cloaker and for the second time when we retrieved the crawler's copy, it behaves like a decent person. It means that cloaker switches between cloaked and non-cloaked mode, so it is difficult to identify cloaking. However, this difference between the first copy of the crawler and the other copies of the crawler and browsers could be because of dynamic nature of web pages.

To find dynamic cloaking exactly, we calculated the term differences between C_1 and C_2 , called T_{CIC2} . If T_{CIC2} is greater than the predefined threshold, there is dynamic cloaking. Results show that about 2.1% of all URLs in our data set belong to this category. If T_{CIC2} is less than the predefined threshold, it means that this difference is only because of dynamic nature of web pages and is not cloaked. Experimental result shows that about 1.6% of URLs in our data set belongs to this category.

- Parameter for identifying cloaking in this step: T_{CIC2} , which means the term differences between the two copies of crawlers.

Several sets of URLs are retrieved from the web then entered to Google search engine to retrieve several sets of URLs. Since spammers try to target pages that have most visitors (popular pages), we decided to find popular searched terms in 2010 from different resources, then entered these queries into Google search engine to get URLs. We used common sources of popular queries. For example, we utilized Google Zeitgeist¹, Yahoo Buzz Index², AOL Hot searches³, Lycos 50⁴, Ask Jeeves⁵ and MSN⁶ to find popular searched terms. We collected about 336 unique queries from these web sites. Then for each query, we retrieved the top 150 responses from the Google search engine. After deleting duplicate URLs, the number of unique URLs is 49632. For each URL, at first, we retrieved two copies; one from the crawler's perspective and one from the browser's perspective. Then we installed Advanced Hash Calculator software and calculated hash value of these two copies. If they are not the same, further copies of crawler and browser are retrieved. The order of retrieving is as follows:

$$C_1 \longrightarrow B_1 \longrightarrow C_2 \longrightarrow B_2 \quad (2)$$

Spammers can identify web crawlers in two ways, IP address and User-Agent HTTP header. For the former one, they keep a list of web crawler's IP addresses and update it frequently. If the IP address of the client is on the list, spammer behaves it like a web crawler. If not, this client is a web browser. For the latter one, web crawler and web browser have different User-Agent HTTP header. Hence spammers can easily detect them. Although, crawler's IP addresses are difficult to fake, User-Agent HTTP header can easily be faked.

In this work we changed User Agent HTTP header to pretend to be a web crawler. We set the User-Agent to be Googlebot/2.1 (+http://www.googlebot.com/bot.html). To pretend to be a web browser, we set the User-Agent to be Mozilla/5, and then downloaded two copies for each of the 49632 URLs, one from the crawler's perspective and one from the browser's perspective. According to our algorithm, if needed, further copies are retrieved, one from the crawler's perspective and one from the browser's perspective. In this work, we only changed User-Agent HTTP header, so pages performing cloaking based solely on IP address will be missed.

Tables 1 – 6 summarize the evaluation results focusing on the following measures:

- Precision: the percentage of cloaked URLs in the URLs predicted to be cloaked
- Recall: the percentage of cloaked URLs in the data set that

¹ <http://www.google.com/press/zeitgeist2010.html>.

² Yahoo's Top Searches of 2010 from <http://buzz.yahoo.com/topsearches2010/lists/>.

³ AOL's Year's Top Searches from <http://about-search.aol.com/hotsearches2010/index.html>.

⁴ Lycos 50 Archives from <http://50.lycos.com/archives.asp>.

⁵ Ask.com's Top Searches from

<http://websearch.about.com/od/topsearches2010/a/topsearchesask.htm>.

⁶ MSN's Top of Live Search for 2010 from <http://blogs.msdn.com/livesearch/>

4. Data Set

are correctly identified

- F-measure: the combination of precision and recall. The F-measure formula is:

$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3)$$

5. Evaluation Results

In this section, we evaluate the performance of the proposed algorithm in details. The performance depends on the precision, recall and F-measure at different threshold values. To evaluate our algorithm, we selected 15% of URLs in each part of the algorithm and checked them manually to determine whether they are cloaked or not. For step 2 we selected about 498 URLs, for step 3 we selected about 275 URLs and for steps 6 and 7 we selected about 387 URLs to check them. Totally we checked 1148 URLs (about 2.3% of all URLs in our data set). For step 5, we selected about 119 URLs (about 10% of all URLs in this part) and checked them manually. The accuracy of our algorithm in this part is about 97.88%. The result of our evaluation is explained in three tables below. For each part of the algorithm we calculated precision, recall and F-measure. In comparison, the performance values of our algorithm in different parts are better than previous methods and there is an improvement in these values. Also our algorithm can find normal pages by retrieving only two copies of a URL. We estimate that about 9% of all URLs in our data set utilized static cloaking and about 2.1% of all URLs utilized dynamic cloaking. Tables 1 to 3 explain our evaluation in different parts.

According to precision, recall and F-value, it is obvious that our algorithm is working more accurate than previous cloaking detection methods. To prove this claim we computed precision, recall and F-value for previously proposed cloaking detection methods (TagDiff2, TagDiff3, TagDiff4 and TermDiff4) and compared them to HashStatic1, HashStatic2 and HashDynamic. The situation for comparison for all methods (our method and previous methods) is the same. It means that we choose the same threshold for all methods, then computed precision and recall. Results have been shown in tables below. Results, based on tables above in predefined thresholds, have been shown in Figure 1.

According to figure 1, our proposed algorithm is working better than previous methods. There is a kind of trade of between precision and recall. In our method, the difference between precision and recall is lower than previous methods, so it can find cloaking web spam more accurate than others. Also our method reaches 1.0000 in both precision and recall in different thresholds. 1.0000 proves the accuracy of the algorithm and based on figure 1, it is obvious that our method

reaches more 1.0000 compared to previous methods.

Table 1. HashStatic1, $f(C_1) \neq f(B_1)$, $f(C_2) \neq f(B_2)$, $f(C_1) = f(C_2)$, $f(B_1) = f(B_2)$

Threshold	Precision	Recall	F-value
1	0.8345	1.0000	0.9070
2	0.8193	0.9794	0.8921
4	0.8574	0.9136	0.8845
8	0.9262	0.8469	0.8847
16	0.9668	0.7864	0.8672
32	0.9894	0.6573	0.7898
64	0.9994	0.5471	0.7999
128	1.0000	0.3675	0.5374
256	1.0000	0.2588	0.4111

Table 2. HashStatic2, $f(C_1) \neq f(B_1)$, $f(C_2) \neq f(B_2)$, $f(C_1) = f(C_2)$, $f(B_1) \neq f(B_2)$

Threshold	Precision	Recall	F-value
1	0.8187	0.9552	0.8819
2	0.8543	0.8749	0.8644
4	0.8977	0.7954	0.8434
8	0.9263	0.7158	0.8075
16	0.9658	0.6647	0.7874
32	0.9552	0.5815	0.7228
64	0.9723	0.4688	0.6325
128	0.9957	0.3864	0.5566
256	1.0000	0.2147	0.3536

Table 3. HashDynamic, $f(C_1) \neq f(B_1)$, $f(C_2) = f(B_2)$, $f(B_1) = f(B_2)$

Threshold	Precision	Recall	F-value
1	0.8438	0.9282	0.8839
2	0.8547	0.8763	0.8653
4	0.8846	0.8175	0.8497
8	0.9225	0.7563	0.8311
16	0.9537	0.6775	0.7921
32	0.9795	0.5834	0.7312
64	0.9879	0.4371	0.6060
128	1.0000	0.2683	0.4230
256	1.0000	0.1776	0.3016

Table 4. Comparison of Precision

Threshold	Hash Static1	Hash Static2	HashDynamic	TagDiff2	TagDiff3	TagDiff4	TermDiff4
1	0.8345	0.8187	0.8438	0.7238	0.8329	0.8576	0.6873
2	0.8193	0.8543	0.8547	0.7963	0.8446	0.8855	0.7536
4	0.8574	0.8977	0.8846	0.8125	0.8754	0.8964	0.7753
8	0.9262	0.9263	0.9225	0.8436	0.9136	0.9136	0.8542
16	0.9668	0.9658	0.9537	0.8795	0.8695	0.9356	0.8863
32	0.9894	0.9552	0.9795	0.9017	0.9475	0.9119	0.9137
64	0.9994	0.9723	0.9879	0.9225	0.9574	0.9645	0.9642
128	1.0000	0.9957	1.0000	0.9413	0.9667	0.9712	0.9738
256	1.0000	1.0000	1.0000	0.9672	0.9713	1.0000	0.9883

Table 5. Comparison of Recall

Threshold	Hash Static1	Hash Static2	HashDynamic	TagDiff2	TagDiff3	TagDiff4	TermDiff4
1	1.0000	0.9552	0.9282	0.9136	0.7736	0.6894	0.7774
2	0.9794	0.8749	0.8763	0.7894	0.7158	0.6323	0.7274
4	0.9136	0.7954	0.8175	0.6856	0.6563	0.5957	0.6971
8	0.8469	0.7158	0.7563	0.6147	0.5773	0.5264	0.6003
16	0.7864	0.6647	0.6775	0.5426	0.5106	0.4327	0.5549
32	0.6573	0.5815	0.5834	0.4637	0.4263	0.3106	0.4793
64	0.5471	0.4688	0.4371	0.3279	0.3164	0.2563	0.3169
128	0.3675	0.3864	0.2683	0.1542	0.1938	0.1438	0.2154
256	0.2588	0.2147	0.1776	0.0996	0.0754	0.0557	0.1663

Table 6. Comparison of F-value

Threshold	Hash Static1	Hash Static2	HashDynamic	TagDiff2	TagDiff3	TagDiff4	TermDiff4
1	0.9070	0.8819	0.8839	0.8077	0.8022	0.7643	0.7295
2	0.8921	0.8644	0.8653	0.7933	0.7747	0.7378	0.7402
4	0.8845	0.8434	0.8497	0.7437	0.7371	0.7158	0.6203
8	0.8847	0.8075	0.8311	0.7112	0.7075	0.6679	0.7050
16	0.8672	0.7874	0.7921	0.6711	0.6433	0.5917	0.6825
32	0.7898	0.7228	0.7312	0.6124	0.5880	0.4624	0.6287
64	0.7999	0.6325	0.6060	0.4838	0.4756	0.4050	0.4770
128	0.5374	0.5566	0.4230	0.2674	0.3229	0.2505	0.3527
256	0.4111	0.3536	0.3016	2.8847	0.1400	0.1671	0.2847

Figure 1. Comparison between our method and previous cloaking detection methods

Figure 2. Time spending to compare crawler and browser's copies with and without Hash function

Figure 2 illustrates the result of the time comparison between the copies of crawler and browser with and without Hash Function. The advantage of introducing Hash Function is to increase the speed of comparison resulting in decreasing required time. To prove the claim, we calculated the time spending to check crawler and browser's copies in different steps with and without Hash Function. In step 1, only one copy of crawler and browser (C1 and B1) are compared. In steps 2, 3, 4 and 5, two copies of crawler and two copies of browser are compared too, but in these steps four comparisons between these four copies will be done. In steps 6 and 7, four copies of a URL (C1, B1, C2 and B2) are retrieved and three comparisons will be done. Results show that by introducing Hash Function the comparison time will be decreased dramatically as shown in figure 2.

6. Conclusion

We proposed an algorithm to find cloaking web spam more accurate than previous cloaking detection methods. This algorithm checks different URLs in different parts to separate cloaked and normal URLs. Experimental results show that our algorithm outweighs previous cloaking detection methods in both precision and recall.

We can improve this algorithm in different ways. This algorithm is only based on term differences between crawler and browser's copies and does not check their links. We can check term differences as well as link differences to make the algorithm more accurate than now. Also in dynamic cloaking detection field, we only checked one situation. There are other situations for detecting dynamic cloaking which we did not address them. For example:

$$C_1 = B_1 = B_2 \neq C_2$$

This situation is also a dynamic cloaking situation, which we did not check it in this paper.

Detecting dynamic cloaking is harder than static cloaking because spammers do not have constant behavior, so it's a good idea for researchers to propose new ways to detect dynamic cloaking.

REFERENCES

- [1] B.Wu, and B.D. Davison. Detecting Semantic Cloaking on the Web, Proceedings of the 15th International Conference on World Wide Web, 819-828, 2006
- [2] Q. Chen, S. N. Yu, and S. Cheng. Link Variable TrustRank for Fighting Web Spam, Proceeding of IEEE International Conference on Computer Science and Software Engineering, 1004-1007, 2008.
- [3] A.Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web Spam with TrustRank, Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), 271-279, 2004.
- [4] B.Y. Pu, T.Z. Huang and Ch. Wen. An Improved PageRank Algorithm: Immune to Spam, Proceeding of IEEE Fourth International Conference on Network and System Security (NSS '10), 425-429, 2010.
- [5] T. Su Tung, and N.A. Yahara. Multi-level Link Structure Analysis Technique for Detecting Link Farm Spam Pages, Proceeding of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, 614-617, 2006.
- [6] Y. Wang, Z. Qin, B. Tong, and J. Jin. Link Farm Spam Detection based on its Properties, Proceedings of IEEE International Conference on Computational Intelligence and Security (CIS '08), 477- 480, 2008.
- [7] B. Wu, and B.D. Davison. Identifying Link Farm Spam Pages, Proceedings of the 14th International World Wide Web Conference, (WWW 2005), 820-829, 2005.
- [8] G.G. Geng, C.H. Wang, Q.D. Li, L. Xu, and X.B. Jin. Boosting the Performance of Web Spam Detection with Ensemble Under-Sampling Classification, Proceedings of IEEE Forth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), 583- 587, 2007.
- [9] M. Mahmoudi, A. Yari, and S. Khadivi. Web Spam Detection based on Discriminative Content and Link Features, Proceedings of IEEE 5th International Symposium on Telecommunications (IST), 542-546, 2010.
- [10] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam Web pages through Content Analysis, Proceedings of the 15th International Conference on World Wide Web, 83-92, 2006.
- [11] S. Ghiam and A. Nemaney Pour. A Survey on Web Spam Detection Methods: Taxonomy, International Journal of Network Security and Its Applications, Vol. 4, No. 5, 119-134, 2012.
- [12] Z. Gyöngyi, and H. Garcia-Molina. Web Spam Taxonomy, Proceedings of the first international workshop on Adversarial Information Retrieval on the Web (AIRWeb '05), 39-47, 2005.
- [13] M.A Najork. System and Method for Identifying Cloaked Web Servers, United States Patent 6910077, 2005.
- [14] B. Wu, and B.D. Davison. Cloaking and Redirection: A Preliminary Study, Proceedings of the first international workshop on Adversarial Information Retrieval on the Web (AIRWeb), 7-16, 2005.
- [15] K. Chellapilla, and D.M. Chickering. Improving Cloaking Detection using Search Query Popularity and Monetizability, Proceedings of the second international workshop on Adversarial Information Retrieval on the Web (AIRWeb '06), 17-24, 2006.
- [16] J. Lin. Detection of Cloaked Web Spam by using Tag-based Methods, Elsevier, An International Journal of Expert Systems with Applications, Vol. 36, No. 4, 7493-7499, 2009.