Cloaking Detection through Simhash-based Website Model and Crowdsourcing

Abstract

Cloaking has long been used by spammers for the purpose of increasing the exposure of their websites, serving as a major malicious technique in search engine optimization. More recently, we have also witnessed a rising trend of employing cloaking in search engine marketing. The motivation of cloaking is to hide the true nature of a website by delivering blatantly different content to users versus web crawlers such as search engines. Cloaking is a popular technique due to its low setup cost and the lack of effective and efficient detection methods.

In this paper, we propose Simhash-based Website Model, or SWM, a new approach to fuzzy hash the content and layout of a website and model the dynamics of a website. We apply SWM to cloaking detection, and our evaluation results show that SWM achieves 97.1% true positive rate with 0.3% false positive rate. SWM is more efficient compared with past approaches and can be deployed as browser plugin for crowdsourcing user views of websites with negligible overhead. In order to understand the incentives behind cloaking, we systematically categorize and review the detected samples. Our analysis shows that majority of them are abusing search engine with traffic sale, and 4.3% are malicious or phishing sites.

Keywords

SEO, SEM, Online Advertising, Cloaking Detection, Similarity Detection Algorithm

1 Introduction

Cloaking is a technique to deliver blatantly different content to users versus censors on the web. Search engine is the main entry for user to get online today, and therefore, attackers who want to promote illegal services or do phishing and malware hosting is willing to have their page ranked high through search engine. On the other hand, search engine develops algorithms to rank pages and penalize these pages. In order to evade detection, attackers use cloaking to hide the true nature of their website, conducting blackhat Search Engine Optimization (SEO). Similarly, in Search Engine Marketing (SEM), advertisers pay search engine for specific advertisements and get their page displayed on the first page of search results. Attackers abuse this service to deliver non-compliant content to user. The foundamental reason of SEO and SEM cloaking is that, search engine is just processing information over the Internet, with no control over it, hence it is hard to tell whether the copy of website that they see are actually what user sees.

In SEO and SEM, attackers use user agent, referer, IP etc based cloaking techniques to avoid inspection. Among the cloaking techniques, IP cloaking is very difficult to detect. There are two major challenges in cloaking detection in SEO and SEM. First challenge is revealing the cloaked content. There are various commercial tools, such as blackjack's, noIPfraud, and wpCloaker [1] that are efficient in performing cloaking. They provide different strategies of cloaking, including user agent, referer, redirect, IP etc. Regarding IP cloaking, they provide services to periodically refresh the list of crawlers' IP addresses. Therefore, revealing cloaked content is an ongoing war between cloakers and censors. Second challenge is differentiating cloaking from naturally dynamic changes of the same page. Dynamic content is allowed in SEO and SEM, however, blatantly deliver different, unrelated content to search engine spider versus normal user is not allowed.

In order to address the first challenge, existing detection approaches pretend to be real user, e.g. set referer, user agent, use multiple IPs or proxy to hide identity. But attackers could identify them and incrementally blacklist IPs. Intuitively, an alternate is to collect data from user side directly. However, it is hard to collect sufficient data for comparison while protecting user privacy. Regarding the second challenge, previous works go through the documents multiple times and compare page difference to

address the second challenge [10, 11, 16, 23]. There are mainly two shortcomings in previous approaches. First, they are slow, because pairwise comparison of documents is computationally expensive and they usually require multiple passes of a document. Second, they are not suitable for crowdsourcing, because they may introduce high overhead to user and violates user privacy.

This work employs simhash to address the two challenges. Charikar's simhash [9] is a special signature of feature set. It is a class of Locality Sensitive Hash (LSH) that has been extensively used in near duplicate in search engine. It is an random projection based algorithm that maps high dimentional data to fixed bits while maintaining the property that, hamming distance between the resulting bits is an estimation of similarity between the original feature set.

We propose Simhash-based Website Model(SWM) to model the dynamics of a web page and use the outlier detection ability of SWM to do cloaking detection. The basic idea is to generate two fuzzy hash corresponding to text and dom tree for each website (fuzzy signature), and compare the differences between spider copy and user copy. In order to model dynamics of a website, we retrieve spider copy multiple times and leverage the fact that hamming distance is euclidean distance (L1 norm) to learn patterns. The proposed system achieves 97.1% true positive rate with 0.3% false positive rate.

By applying the trained model on collected dataset, we detected 2600 cloaking samples in total. Then, we manually classify and illustrate the incentives behind cloaking, results show that the majority are doing blackhat SEO for traffic sale, 4.3% of cloaking sites are malicious download or phishing sites. A breakdown of the results are described in §5.

With motivation of both search engine and user to combat cloaking, we propose a novel model to collect simhash from user side, with low overhead and privacy guarantee provided by RAPPOR. By crowdsourcing the fuzzy signatures of what is actually shown to user, we address both challenges in cloaking elegantly.

The remainder of this paper is structured as follows. Section §2 provides a technical background on cloaking and simhash, and related work in cloaking detection. Section §3 introduces methodlogies, including simhash-based website model, cloaking detection algorithm and dataset collection. Followed by evaluation of our model in Section §4. Section §5 explains our detection result in SEO and SEM and motivate the deployment. Section §6 compares our work with past approaches and describes server-based and crowdsource-based deployment, proposes the frameworks and corresponding pros and cons. Section §7 concludes this work.

2 Background

Search engine, as a most innovative technology introduced in late 20th century, has widely influenced our relationships. Search engine used their own page ranking algorthim to rank and indexed websites. In this way, users could find information by entering the search terms into the search enginee. To get better rankings and accurate indexed on search engines such as Google, websites are encouraged by Google to optimize their content. This is a technology called search engine optimization (SEO). SEO policies were introduced by search engine later to maintain a fairness of search enginee ranking. Recently, some websites break the SEO policies to get better rankings on websites because of the large profits. These technology are called Black Hat SEO, which is used to get better rankings on search engine but break the SEO policy. From the SEO policy, one of the most efficient methods to get better page rankings is to update the content of website frequently [23]. Following this rule, cloaking, a Black Hat SEO method, is invented. Cloaking serves a blatently different information to users and Google to maintain their profits and get better page rank. To be clear, we give a general example. Websites provide a frequently updated information to the Google so that they could get better rank. On the other side, they send a different information to users to get profits.

2.1 Example

To concrete the idea of cloaking, a specific cloaking example, traffic sale cloaking, is introduced. We entered the term "instant loan quote" in Google. Google returned a set of search results as showned in Figure 7(a), and the third result is doing cloaking. When clicking this link directly, Figure 7(b) is presented. However, if we visit the landing page as spider (set user agent to Google bot), Figure 7(c) is shown. The fact is that, this website is doing blackhat SEO, raising its rank with junk content, but when user clicks, they simply iframe another page, monentizing from its ranking and the traffic.

2.2 Search Engine Marketing

Priviously, we briefly talked about search engine optimization. In this section, we introduce another area called search engine marking(SEM) where enourmous cloaking websites exist. According to wikipedia [6], the term SEM is used to mean pay per click advertising, particularly in the commercial advertising and marketing communities which have a vested interest in this narrow definition. To be clear, search egnine marketing defines the advertisements shown on search engines after searching terms. Accoding to wikipedia [6], boundary beween SEO and

SEM are sometimes not clear. However, when studying the cloaking, the SEO and SEM are clearly different. The difference betwen them is because that the policies and rules on SEO and SEM are different. On SEM, the policy are usually strict and commertial-related. Further, the cloaking-incentives on SEO and SEM are different. On SEM, websites are more likely to provide illegal medicines and services. On SEO, websites are inclined to do malicious downloading and phishing. Further, the strategy on cloaking detection on SEO and SEM are different. The difference is crucially decided by ranking mechanisims. In SEO, though the page rank algorithm is not fully public, it is well known that rankings are related to content, page rank and visit traffic. In SEM, the rank algroithm depends on real time biding system. Websites need to bid the "pay per click" price on the real time biding system. The one with the higher price will be shown in higher priority. The terms with large commerical potential ask higher price. Because of the difference between SEM and SEO, the strategy to detect cloaking should change. The cloaking detection algorithm should change such as collecting commercial related terms, crawling adverstisements, checking the SEM policy(Google Ads Policy) [2]. To ensure "Pay per click" mechanisms, in other words, websites won't pay much extra money because of cloaking detection, we introduced a new model and "click counting" mechanisms. Traditional methods failed to do this adjust on SEM. Thus, the traditional methods are inefficient detect cloakings in SEM.

2.3 Cloaking Types

To understand how cloaking works in different scenarios, we will discuss the cloaking types. In order to serve targeted users cloaking content, the scammer must use some identifiers to distinguish user segments. Based on these used identifiers, the cloaking techniques are classified as Repeat Cloaking, User Agent Cloaking, Referred Cloaking and IP Cloaking.

In Repeat Cloaking, websites store the visit history in user-end(Cookies) or server-end (server log). Based on visit history, the website presents different information. According to [23], they oberved websites that only show cloaking at first time, in the hopes of making a sale, but subsequent visits are presented with benign page. In our observation, some repeat cloaking websites are willing to show the same content as the first time. In User Agent Cloaking, websites check the User Agent Field in the HTTP request. From the User agent field, they could find the crawlers that uses the well-known User-Agent strings and identify crawlers. In Referrer Cloaking, websites examamine the referer field of HTTP request. From the referer field, the website could easily find if the users clicked though search engines to reach their websites. In this way, the cloaking websites only serves the scam page to the targeted users from search engines. In IP Cloaking, websites determine the visitors' identities by their IP addresses. With an accurate mapping between IP addresses and ornizations, the websites could easily distinguish cralwers from search engines and real users.

2.4 Previous Work

2.4.1 Cloaking Detection

The major challenge in claoking detection, is to differentiate dynamic pages from blatantly different content. Comparing document word by word is very expensive and slow, therefore, various ways to test similarity of documents are proposed. [14] considered cloaking as one of the major search engine spam techniques. [18] proposed a method of detecting cloaked pages from browsers by installing a toolbar. The toolbar would send the signature of user perceived pages to search engines. [26] use statistics of web pages to detect cloaking, [10] detected syntactic on the most popular and monetizable search terms. They showed that monetized search terms had a higher prevalence of cloaking than popular terms. Referrer cloaking was first studied by [24]. They found a large number of referrer cloaking pages in their work. [16] use tag based methods,

[23] extended their previous efforts to examine the dynamics of cloaking over five months, identifying when distinct results were provided to search engine crawlers and browsers. They used text-based method and tag-based method to detect Cloaking page. [11] use summarize previous work and compare text, tag, link based approaches. However, JavaScript redirects were not able be handled by their crawler.

However, non of them take into consideration the data collection part. Nor do they take into account the efficiency of the algorithms.

In order to detect cloaking, we introduce Simhash-based Website Model (SWM) and design and implement a much more efficient algorithm based on SWM, which has comparable false positive rate and true positive rate. We take into account the efficiency of the algorithm and implement a plugin which introduce negligible overhead to user. With crowdsourcing, we can solve cloaking at scale.

2.4.2 Simhash

Charikar's simhash [9] has been widely used in near duplicate detection in search engine. [13] conducted a large-scale evaluation of simhash against Broder's shingle-based fingerprints [8] in finding near-duplicate web pages. A great advantage of using simhash over shingles is that it requires relatively small-sized fingerprints. For example,

shingle-based [8] requires 24 bytes per fingerprint. In comparison, [17] shows that for 8B web pages, 64-bit simhash fingerprints suffice.

These approaches are mainly focusing on using simhash to signature websites on the Internet and comparing them all together to identify near duplicates. In contrast, this work leverages the fact that content from same url is supposed to be near duplicate and employs simhash to do outlier detection.

This work adopts the same simhash algorithm and setting as [17], but with different method of extrating text features. [17] extracts text features from website with standard IR techniques, and weigh each feature with inverse document frequency. This requires extra information and introduces overhead if deployed in browser at user side. In §3, we describe our approach of extracting text features. Inspired from previous cloaking work [23] which compares both text and dom tree information, we extract dom features as well.

3 Methodology

As mentioned in §2, there are two challenges in cloaking detection: reveal the content and handle dynamics of a webpage.

In order to model the dynamics of a webpage, we propose Simhash-based Website Model, that is, use clusters learned from fuzzy hashs to model average and variance of each change. This is based on the assumption that, the content and layout of a website delivered to different users each time, is consistent despite of the dynamic part on the page, e.g. advertisements. In order to reveal the cloaked content, we design and implement simhash algorithm as browser plugin, enabling crowdsourcing from user side. Besides, we demonstrate the complexity of simhash algorithm, and show that this plugin introduces negligible overhead to user browser.

3.1 Simhash-based Website Model

A website is usually rendered through Document Object Model (DOM), which is maintained by browser in the fashion of a tree. DOM tree contains information about layout of a website (Cascading Style Sheets (CSS) is supplemental to DOM in describing the look and formatting of a document). Out of various kinds of DOM nodes, text nodes represents the actual text that is displayed to user. This work focuses on structure of DOM tree and text nodes that user actually sees, and use simhash clusters to model them.

3.1.1 Distance Approximation

Simhash [9] is a dimensionality reduction technique. It is a fuzzy hash function family that maps a high dimension dataset into fixed bits and preserves the following properties: (A) The fingerprint of a dataset is a "hash" of its features, and (B) The hamming distance between two hash values is an estimation of the distance between the original datasets. This is different from cryptographic hash functions like SHA-1 or MD5, because they will hash two documents which differs by single byte into two completely different hash-values and the hamming distance between them is meaningless. In constrast, simhash will hash them into similar hash-values.

3.1.2 Computation

In terms of simhash computation, we use the same settings described in [17], which turns out to be effective given the corpus of 8 billion websites over the Internet.

The computation of simhash start from a set of features. Given a set of features extracted from a document and their corresponding weights, we use simhash to generate an f-bit fingerprint as follows. We maintain an f-dimensional vector V, each of whose dimensions is initialized to zero. A feature is hashed into an f-bit hash value. These f bits (unique to the feature) increment/decrement the f components of the vector by the weight of that feature as follows: if the i-th bit of the hash value is 1, the i-th component of V is incremented by the weight of that feature; if the i-th bit of the hash value is 0, the i-th component of V is decremented by the weight of that feature. When all features have been processed, some components of V are positive while others are negative. The signs of components determine the corresponding bits of the final fingerprint. In this work, we set f to 64.

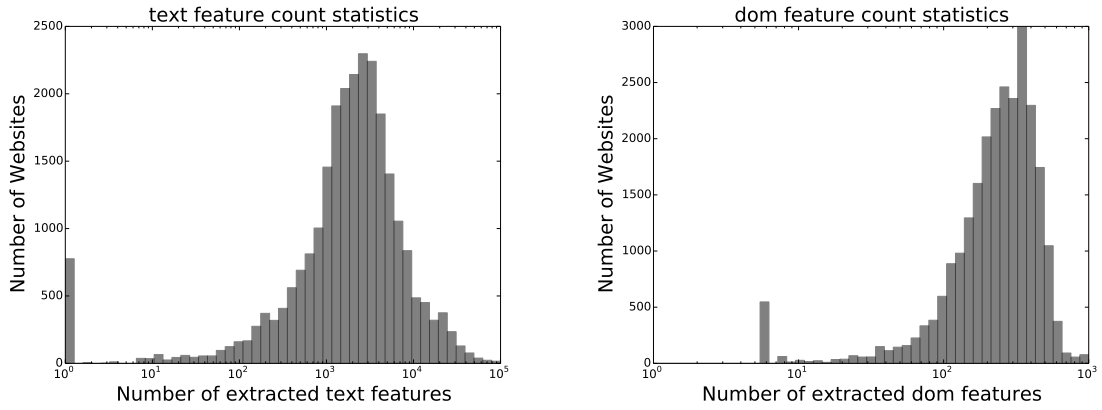
3.1.3 Requirements On Feature Set

There are two characteristics in the computation of the simhash. First, the order of the features doesn't matter because simhash is maintaining a global counter V. Second, size of the feature set should be relatively large, because simhash is random projection based approach, small set of features may result in completely different simhash with one feature difference due to randomness.

The two characteristics can be considered as requirements in feature selection phase: (A) If the order of feature matters, feature set should include structural information. (B) Size of feature set should be relatively large.

3.1.4 Text Simhash and DOM Simhash

In order to detect cloaking, we need to capture the bahavior and similarity that a same website maintains. Inspired



(a) (b)

Figure 1: Number of text and DOM features extracted from different websites

by [23], this work looks at both text and DOM features and generate simhash separately.

For text simhahsh, the algorithm extracts visible sequence of words from website. From the sequence, this algorithm extracts words, bi-gram, tri-gram set (repeated elements only recorded once). For example, for sentence: thank you so much, corresponds to feature set {thank, you, so, much, thank you, you so, so much, thank you so, you so much}. Because there are usually large number of words on a website, Requirement (B) suffices. Figure 1(a) shows text feature statistics on urls obtained from hot search results $D_{hot,search}$. Since bi-gram, tri-gram shows how words are concatenated and represents strcture of documents, Requirement (A) suffices. This algorithm is used rather than weighted version described in [17], because it is light-weight and doesn't require inverse document frequency and phrase detection. This is important in our case, because we are designing an algorithm that is deployable in user browser, where efficiency matters.

Regarding structure of websites, we design an algorithm to extract DOM features out of DOM tree.

For each DOM tree, we record presence of each node (tag name), as well as presence of each child parent pair. The node set tells us information about what tag is present in this page, and child parent pair tells us how these tags are organized, i.e., structure information, suffices Requirement (A). Since we are recording presence of tags and type of tags are relatively small, we record tag name and associated attribute names to gain more features (higher entropy). Attribute value is discarded because based on our observation, attribute value may change on every visit. For example, Figure 2 correspond to feature set {html, head, body, title, ..., (head, html), (body, html), ...}

Figure 1(b) shows DOM feature statistics $D_{hot,search}$ and the number of features extracted in this fashion is relatively large, suffices Requirement (B).

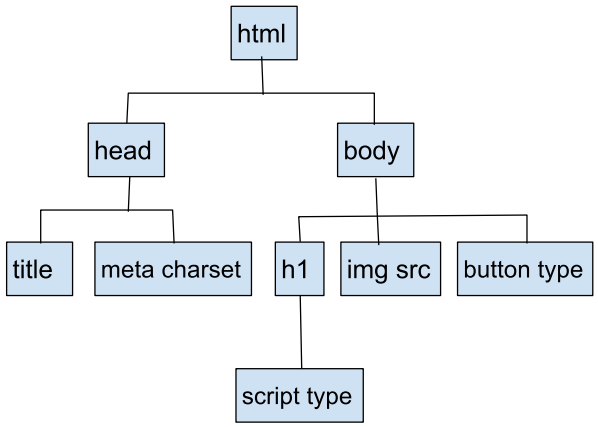
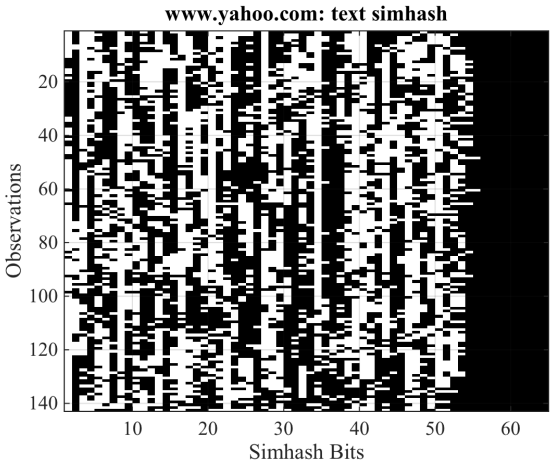


Figure 2: DOM tree example

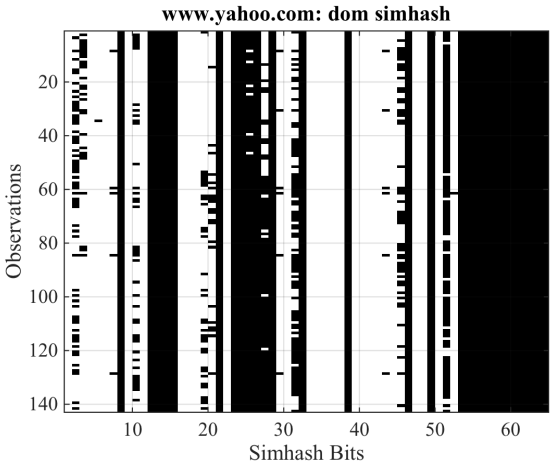
Figure 3 gives an example on how DOM and text simhash looks like and how they are distributed on *www.yahoo.com* over 7 x 24 period from Feb.1, 2015 to Feb.7, 2015. Figure 3(a) shows text simhash and Figure 3(b) shows DOM simhash. The x-axis is bits of simhash value and y-axis is the id of each observation (id increases in the order of collection time).

It is pretty straightforward from Figure 3 that, text simhash changes rapidly, indicating dynamic nature of this website, and DOM simhash changes relatively slow and less.

Till now, we have demonstrated the algorithm we are using to generate text-simhash and DOM-simhash out of a website. Based on our observation, the text-simhash might change rapidly, while DOM-simhash relatively remain the same.



(a) User View of Yahoo Text Simhash



(b) User View of Yahoo DOM Simhash

Figure 3: Yahoo simhash changes over 7x24 period Feb.1-7, 2015

3.1.5 Clustering

Websites could have multiple versions of content due to many reasons. An example is page construction. The server might return some notice or help information for apology and instructions on what to do next. Since we are monitoring websites over a period of time, we need to be able to build model for different version of websites and learn corresponding pattern.

Different from [17], we not only want to know whether two pages are duplicate, we also want to know the patterns of these simhash. In this work, we employ hierarchical clustering to do this job.

Since simhash maps a high-dimensional features set into fixed number of bits, where hamming distance represents the similarity between the original feature set. We leverage the fact that hamming distance is euclidean distance (L1 norm on 64 dimension). On each dimension, the original value is either 0 or 1. But it can be averaged and centriod can be computed.

In this work, we use agglomerative hierarchical clustering [15] to cluster collected simhash. This is a bottom up approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. In order to decide which clusters should be combined, a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate metric (a measure of distance between pairs of observations), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets. This work represents each simhash as a 64 dimension bit vector and specifies hamming distance as distance metric. The linkage method used is average distance and criterion is inconsistent coefficient.

Average linkage: Consider the following use case, spider collect multiple copies of a website and compute corresponding simhash $S_{spider} = \{s_i, i \in (1,n)\}$ and the user return observation s_{user} for this website. In order to compare s_{user} with S_{spider}, and take into consideration of the all the collected simhash, the distance from s_{user} to centroid of S_{spider} is a reasonable measure. In order to be consistent with detection phase (user versus spider), clustering phase uses average linkage as well.

Inconsistency coefficient: this coefficient characterizes each link in a cluster tree by comparing its height with the average height of neighboring links at the same level and below it in the tree. The higher the value of this coefficient, the less similar the objects connected by the link. By using threshold of inconsistency coefficient as criterion, we could get several clusters for each website.

$$\alpha = \frac{d-\mu}{\sigma} \tag{1}$$

Equation 1 explains how inconsistent coefficient is computed. α is inconsistent coefficient, d is the distance between two clusters, μ is mean of the heights of all the links included in the calculation, σ is standard deviation of the heights of all the links included in the calculation. In our case, each computation includes all links at the same level and below it, and this is done by setting $depth$ in $inconsistent(Z, depth)$ [5] to $m-1$, where m is total number of observations.

Let T_{learn} denote the threshold for inconsistent coefficient, Equation 2 shows the merging criterion in clustering phase. After clustering, S_{spider} is divided into c clusters $S_{spider,k}, k \in (1,c)$. These clusters are Simhash-based Website Model for this website.

We denote each cluster $S_{spider,k}$ with centroid and links formed in clustering phase $S_{spider,k} = \{centroid, links\}$. This representation is intended for comparison with a

new observation (single node cluster). *centroid* is used to compute distance from new observation to current cluster, *links* are used to compute μ and σ.

$$\begin{gathered}\alpha = \frac{d_{r,s}-\mu}{\sigma}, \text{merge } S_{spider,r}, S_{spider,s} \text{ if } \alpha < T_{learn} \\ \text{where } S_{spider,r} = \{s_{spider,r,i}, i \in (1,n_r), links_r\}, \\ S_{spider,s} = \{s_{spider,s,j}, j \in (1,n_s), links_s\}, \\ d_{r,s} = \frac{1}{n_r n_s}\sum_{i=1}^{n_r}\sum_{j=1}^{n_s} dist(s_{spider,r,i}, s_{spider,s,j}), \\ \mu = avg(x), x \in links_r \cup links_s, \\ \sigma = std(x), x \in links_r \cup links_s\end{gathered} \tag{2}$$

3.2 Cloaking Detection

In the above section, for observations of each website, S_{spider}, we have learned clusters , which is $S_{spider,k}, k \in (1,c)$. For each cluster, we represent it as $S_{spider,k} = \{centroid, links\}$. In order to use these clusters, we first illustrate how the comparison can be done.

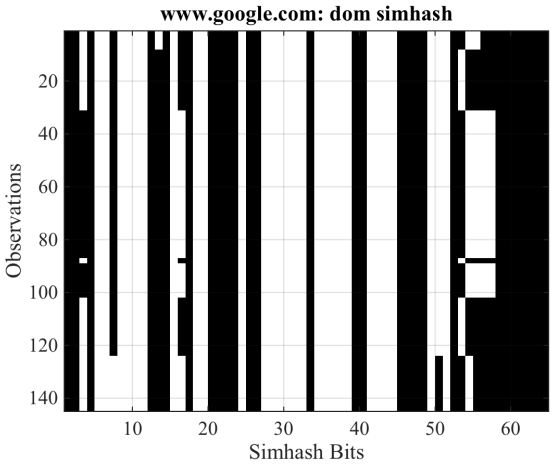
In order to detect SEO and SEM cloaking, this work first search and click results with normal user agent, then visit landing urls collected with google bot user agent (described in detail in §3.3). For a specific website, we denote user observation of this website as s_{user}, denote spider copies as S_{spider} and $S_{spider,k}$ are the learned clusters.

For each cluster $S_{spider,k}$, we record link heights and centroid. When comparing s_{user} with $S_{spider,k} = \{centroid, links\}$, distance d' from s_{user} to *centroid* is computed, which is average distance from cluster $\{s_{user}\}$ to $S_{spider,k}$. In §3.1.5, we merge clusters if inconsistent coefficient $\alpha < T_{learn}$. Similarly, we could compute inconsistent coefficient α' for $\{s_{user}\}$ to $S_{spider,k}$ as shown in Equation 3 and use detection threshold to reject s_{user}.

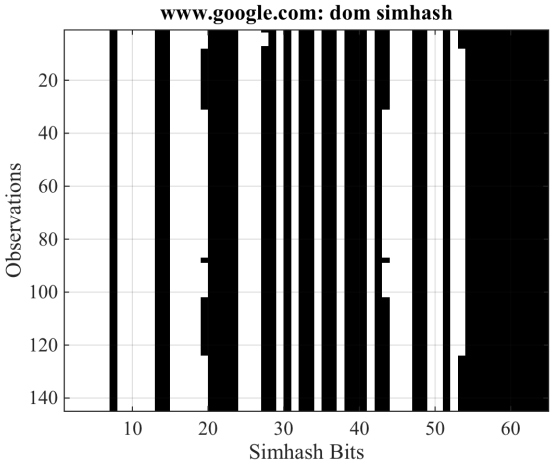
$$\begin{gathered}\text{If } \alpha' = \frac{d'-\mu}{\sigma} > T_{detect}, \text{reject } s_{user} \\ \text{where } S_{user} = \{s_{user}\}, \\ S_{spider,k} = \{centroid, links\}, \\ d' = \frac{1}{n_k}\sum_{i=1}^{n_k} dist(s_{user}, s_{spider,k,i}) = dist(s_{user}, centroid), \\ \mu = avg(x), x \in \{d'\} \cup links, \\ \sigma = std(x), x \in \{d'\} \cup links\end{gathered} \tag{3}$$

For observation s_{user}, if all clusters $S_{spider,k}, k \in (1,c)$ rejects it, then mark s_{user} as cloaking.

However, in reality, there can be consistent differences between what spider sees and what user sees and this is totally leagal. For example, a website delivers non-javascript version to spider, but javascript version to user. Figure 4 shows the difference between spider and user observations of *www.google.com* DOM simhash. Another



(a) User View of Google DOM Simhash



(b) Spider View of Google DOM Simhash

Figure 4: Comparison of user and spider copies DOM simhash, over 7x24 period Feb.1-7, 2015

example is that a website may present advertisements to normal user, but non-advertisement version to spider. Besides, there are websites that rarely changes at all, meaning σ is zero, thus Equation 3 doesn't work. To fix the two problems, we introduce a minimum radius R_{detect}. The modified formula is Equation 4:

$$\text{If } d' - R_{detect} - \mu > T_{detect}\sigma, \text{reject } s_{user} \tag{4}$$

Next section §4 evaluates the proposed SWM and cloaking detection algorithm, and provides insights and suggestion on selection of $T_{learn}, T_{detect}, R_{detect}$.

3.3 Dataset

This work is focusing on detecting cloaking in SEO and SEM, therefore, we collect search words for both SEO and SEM.

3.3.1 Keywords

Similar to [23], in order to detect and measure cloaking that intended to gather high volumes of undifferentiated traffic, and those target on specific cloak search terms, we collected two set of words, hot search words $W_{hot,search}$ and abuse oriented words $W_{spam,search}$. $W_{hot,search}$ consists of 170 unique monthly hot search words from Google trend [4] from Jan 2013 to Dec 2014. $W_{spam,search}$ is first manually collected by referring to [2] for basic abuse oriented words in search engine, mainly from categories including gaming ad network, adult-oriented content, alcoholic beverages, dangerous products, dishonest behavior, gambling-related content, healthcare and medicines. Then we expand $W_{spam,search}$ using Google Suggest, and get 1024 words.

For SEM cloaking detection, the selection of keywords is a different, because Google Adwords system is based on a real time bidding system and advertisers will bid on monetizable keywords, e.g. instant loan, but not on navigational keywords, e.g. facebook. Inspired from [10], we collect monetizable words for advertisement collection. Monetizability is measured by Google Keyword Planner (GKP) [3]. Keyword planner provides convenient API for checking competition and suggested bid for list of keywords. Similar to SEO, we collect monetizable keywords from hot search and abuse oriented list. Abuse oriented ad words are collected by filtering words that has no competition and no bid price in $W_{spam,search}$, as a result, 573 words composes spammy advertisement word set $W_{spam,ad}$. In order to get as much advertisements as possible, we collect 11671 hot keywords from Google Trend [4] (from 2004 to 2014, and each categories are collected), filter this word set by GKP, and get 4108 keywords, $W_{hot,ad}$.

3.3.2 Crawling

Starting from the four collected keyword set, we automate browser to do search-and-click. This work uses Selenium [21], an open source browser automation tool, to visit websites while mimicked as users and spiders. For SEO keywords crawling, we set browser user agent to

Googlebot/2.1 (+http://www.google.com/bot.html)

to mimic Google bot and

Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2049.0 Safari/537.36

to mimic Chrome user on windows machine. For SEM keywords, Chrome user setting is the same, but spider user agent is set to *AdsBot-Google (+http://www.google.com/adsbot.html)*, which is the user agent used by Google to inspect ads and advertisement landing pages are required by Adwords Policy [2] to show consistent content to Google and normal user.

In SEO, we use a university IP running ubuntu to perform crawling (browser user agent can be forged, thus will be considered as windows machine by site owner). Since redirection is popular on Internet, what user actually clicks may not be where he goes to. In this work, we denote the link that user directly clicks on or input to browser as URL_{visit}, and resulting page that user is finally led to as $URL_{landing}$. The crawling process is: (A) For each word in $W_{hot,search}$ and $W_{spam,search}$, we search in Google and click on top 200 result as normal user. The landing url $URL_{landing,user}$ and website content are saved to disk. (B) For landing urls collected in step (A), we directly visit them 6 times (because we need multiple spider copies to learn clusters) as Googlebot and save website content and record the landing url in step (A), $URL_{visit,spider} = URL_{landing,user}$.

The above steps are different from past approaches [16, 25, 23], which visit $URL_{visit,user}$ in step (B). Our approach is reasonable, because we leverage the fact that, in order to reach real user, $URL_{landing,user}$ have to show cloaked content to user. If site owner cloaks on landing page, we can catch them; and if site owner employs redirect cloaking [25], he has the incentive to cloak on landing page in order to evade inspection.

In SEM, the crawling process is similar, except that: word sets are $W_{hot,ad}$ and $W_{spam,ad}$; click and visit ads in fist 5 pages as normal user (empirically only first 5 pages contains advertisemetns); and spider user agent is set to ads bot.

3.3.3 Data Statistics

Through steps described in §3.3.2, we get four datasets, $D_{spam,search}$, $D_{hot,serch}$, $D_{spam,ad}$ and $D_{hot,ad}$. Since we are modeling website on a per url basis and parameters in url may change every time of visit, it is necessary to define the granularity of comparison. For example, ad campaign information are encoded in the landing url of many advertisements, which is different almost every visit, making it difficult to compare based on exact url. For simplicity, we strip all the parameter values and keep all the parameter names, and throw away the scheme field, i.e. *http://www.gatech.edu/?user=1234* is simplified to *//www.gatech.edu/?user=*. Under this definition of url, $D_{spam,search}$ has 129393 unique urls, $D_{hot,search}$ has 25533, $D_{spam,ad}$ has 2219, and $D_{hot,ad}$ has 25209 [1].

4 Evaluation

In the above section §3, we propose the SWM and explains how it can be used to do cloaking detection (outlier

[1] Automating website visit through selenium and chrome can sometimes result in rendering errors or empty content, these examples are removed from collected dataset

detection). There are three parameters to be learned, the upper bound of inconsistent coefficient in clustering phase T_{learn}, the lower bound of inconsistent coefficient in the detection phase T_{detect}, the fix parameter minimum radius R_{detect}. In order to measure cloaking in both SEO and SEM, we collects four candidate dataset §3.3. In this section, we first describe the groundtruth obtained from $D_{hot,search}$ and $D_{spam,search}$, then use it to train and test the performance of the proposed model.

4.1 Groundtruth

Similar to [16], we start by remove duplicates (same simhash from user view and Google view) from $D_{hot,search}$ and $D_{spam,search}$, because these are not helpful for the algorithm training (designed to handle dynamics of websites, non-changing websites are handled by default). Then we manually label websites from $D_{hot,search}$ and $D_{spam,search}$ using heuristics such as domain reputation [7] until we have relatively large sample size for training. It is important to notice that, although the labeling process uses some url reputation information (highly reputated domains are less likely to do cloaking), it is not relevant to the algorithm, for algorithm only measures text difference and layout difference of the original documents. By conducting this massive labeling process, we collect 1195 cloaking examples. In terms of normal websites, we randomly select 5308 samples from non-cloaking dataset. The two parts, 6503 urls in total, are combined as groundtruth D_g for algorithm training and evaluation. By applying feature extraction and simhash computation described in §3.1, we have each url associated with its DOM simhash $S_{g,DOM}$ and text simhash $S_{g,text}$.

4.2 Detection and Evaluation

Regarding parameters to learn, T_{learn} and T_{detect} are parameters to handle page dynamics, and R_{detect} is a fix parameter to make system robust to consistent difference between spider and user copies, which doesnot contribute to website modeling. Therefore, we first select optimal T_{learn} and T_{detect}, and then present system performance for different settings of R_{detect}.

4.2.1 Selection of T_{learn} and T_{detect}

Because R_{detect} is a parameter to allow the system to handle consistent difference between spider and user copies, therefore, we first set detect R_{detect} to be zero, and do five-fold stratified cross validation [19]. on with groundtruth D_g. In the learning phase, our objective function is to first minimize the total number of errors in classification $E = FP + FN, FP =$ false positive, $FN =$ false negative, and if E is the same, minimize $d = T_{detect} - T_{learn}$. This

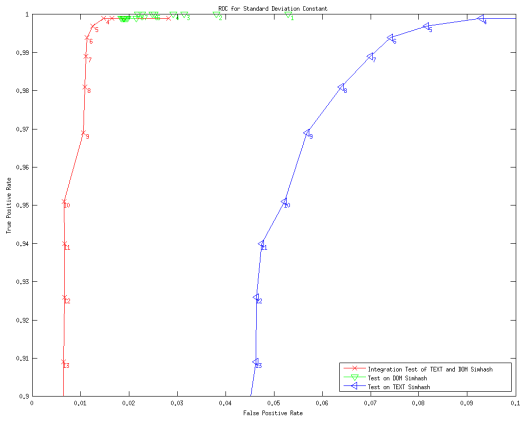


Figure 5: ROC for DOM, TEXT, DOM & TEXT

is reasonable because d is the area that we cannot reject or accept. The smaller the area, the more compact the learned model.

By applying five-fold stratified cross validation on $S_{g,DOM}$ and $S_{g,text}$ and the described objective function for optimal parameter selection, DOM simhash returned $T_{detect,DOM} = 1.8$ and $T_{learn,DOM} = 0.7$, and text simhash yields $T_{detect,text} = 2.1$ and $T_{learn,text} = 0.7$.

4.2.2 Radius Selection R_{detect}

Similarly, $R_{detect,text}$ and $R_{detect,DOM}$ are decided separately. In this section, we conduct three experiments: cloaking detection using (1) DOM simhash (2) text simhash (3) intersection DOM simhash and text simhash result. Again, we use five-fold stratified cross validation and same objective function as in §4.2.1 to learn and test $S_{g,DOM}$ and $S_{g,text}$. The optimal parameter for DOM simhash is $R_{detect,DOM} = 17$, text simhash is $R_{detect,text} = 16$. Figure 5 gives an illustration on how False Positive Rate (FPR) and True Positive Rate (TPR) changes as threshold for DOM and text changes.

Next, in order to show the combined result for different settings of $R_{detect,dom}$, $R_{detect,text}$, we set $R_{detect,text}$ around its optimal value and change $R_{detect,dom}$ as shown in Figure 5. It is obvious that combining both DOM and text features improved the performance. The learned parameters are used in §5 to detect cloaking on the four dataset, and since we want to capture as much cloaking as possible for incentive study, we choose $R_{detect,text} = 15$ and $R_{detect,DOM} = 13$, which corresponds to 0.3% FPR and 97.1% TPR.

Category	Traffic Sale				PPC	Error	IS	Phishing	PD	Malware	Total
	Pharmacy	Gamble	Loan	TS							
Spammy Search	661	1514	33	28	28	43	122	17	73	20	2491
Hot Search	33	2	26	27	0	2	2	0	3	0	93
Spammy Ads	0	0	0	0	1	0	5	0	0	0	6
Hot Ads	0	0	0	4	0	0	6	0	0	0	10

TS: Traffic Sale, PPC: Pay-Per-Click, IS: Illegal Service, PD: Parking Domain

Table 1: Cloaking Distribution.

5 Measurement

With the model built in §4, we detect cloaking in four collected datasets, spammy search, $D_{spam,search}$, hot search, $D_{hot,search}$, spammy ads $D_{spam,ad}$, hot ads, $D_{hot,ad}$. $D_{spam,search}$. From our observations, we categorize cloaking websites into seven types: Traffic Sale, Pay-Per-Click (PPC), Error Page, Illegal Service, Phishing, Parking Domain and Malware Downloading. To better analyze cloaking incentives, we divided traffic sale into four categories: pharmacy, gambling, loan and general traffic sale. Traffic sale sites are usually third party url composed of single iframe pointing the target site for traffic monetization, Figure 7 gives an example. PPC means user view of landing page simply host pay-per-click advertisements. Error Page refers to website that deliver simple information, such as character 'P' to user, but SEO content to Google, the reason is unknown. Illegal Service includes websites that provide essay writing service, copyrighted content and bot service, e.g. buy 10k youtube likes. Parking Domain refers to domains that redirect user to unwanted download, but not necessarily malicious.

5.1 Cloaking in SEO

In SEO, we detect cloaking websites in $D_{spam,search}$ and $D_{hot,search}$. In $D_{spam,search}$, we applied our cloaking detection system on 129393 websites. We manually label the detection results and identified 2491 cloaking websites which are further labeled with its incentive Table 1. Majority of these sites fall into traffic sale. But it is worth noticing that phishing, parking domain and malware sums to 110 in the table, which need to be cautious about. In $D_{hot,search}$, cloaking detection system returned 93 cloaking websites. 33 websites are cloaking of pharmacy. 2 websites are cloaking of gambling. 26 websites are cloaking of loan. 27 websites are cloaking of traffic sale. 2 websites are cloaking of illegal service. 3 websites are cloaking of parking domain.

From the detection result, we see that the main goal of cloaking as an SEO technique is to obtain user traffic. In spammy search, 89.7% is traffic sale cloaking. 97.27% of them are from pharmacy and gambling. In hot search, 94.6% is traffic sale cloaking. We conclude that majority of cloaking websites in SEO field is doing traffic sale, but there are sites promoting illegal service and even phishing and malware as well.

5.2 Cloaking in SEM

In SEM field, we deteck cloaking websites in $D_{spam,ad}$ and $D_{hot,ad}$. In spammy ads, we applied cloaking detection system on 25533 websites. Cloaking detection system reported 6 cloaking websites. 1 website is cloaking of pay per click. In hot ads, we applied cloaking tection system on 25209 websites. Cloaking detection system reported 10 cloaking websites. 4 websites are cloaking of traffic sale. 6 websites are cloaking of illegal service.

In spammy ads and hot ads, none of cloaking websites are traffic sale cloaking. One reason is page ranking mechamisms in SEM. Because most of ads are ranked by their real time bid price, there is no incentives to raise page rank by increasing traffic sale. In addition, using ads to increase traffic sale can cost a lot. Instead of using ads to increase page rank, a direct way is to pay search engine like Google to get higher rank. Most of cloaking websites in SEM have strong commercial incentives. In spammy Ads, 83.3% are providing illegal services to gain profits. 16.7% are pay per click cloaking website. As long as the pay per click cloaking website get enough clicks and gain more money than bidding cost, it is worthy to do pay per click cloaking in SEM. In hot ads, we see 60% cloaking websites are providing illegal service. This fact confirmed our conclusion that most of cloaking in SEM are motivated by commercial profits.

6 Discussion

In this section, we first compare our work with previous cloaking detection works, then discuss two kinds of deployment of simhash-based cloaking detection system: server-based and crowdsoucing-based. We analyze the attack models and robustness of crowdsourcing model.

W: number of words, T: number of tags, L: number of links					
Methods	Features	Complexity	Efficiency	Effectiveness	Deployment
Najork [18]	words, links, tags	$O(W^2+L^2+T^2)$	low	low	User
Term & Link Diff [25]	words and links	$O(W^2+L^2)$	low	medium	Server
Wu & Davison [26]	words, links	$O(W^2+L^2)$	low	medium	Server
CloakingScore [10]	words	$O(W^2)$	low	medium	Server
TagDiff [16]	tags	$O(T^2)$	medium	medium	Server
Cloaker and Dagger [23]	words, tags	$O(W+T^2)$	medium	high	Server
Hybrid Detection [11]	words, links, tags	$O(W+L^2+T^2)$	medium	high	Server
SWM	words, tags	$O(W+T)$	high	high	Server and User

Table 2: Comparison of cloaking detection methods

6.1 Efficiency Comparison

Previous cloaking detection approaches use different feature sets from a website, and the latest work [23] use text features, dom features and search snippet to detect cloaking. Comparing with their model, we achieve similar precision and recall (transformed from FPR and TPR). However, a great advantage of our work over past approaches is that, our algorithm is efficient and clean: requires only single pass of website to get fuzzy signature and then compare fuzzy signatures instead of the original document. This feature not only saves time and make cloaking detection scalable, but also enables data collection to be deployed at user side for crowdsourcing.

Table 2 compares past cloaking detection approaches. Our approach use the content that contains most information (entropy), and is more efficient compared to past approaches. Regarding text simhash and DOM simhash generation, we implement a browser plugin and it introduces little overhead to browsing session. Besides, in order to further reduce overhead to browser, simhash computation can be triggered under some circumstances. For example, compute simhash only when advertisements are visited (url contains specific string), and set computation probability to $p = \frac{1}{100}$ to further reduce overhead.

6.2 Server-based Deployment

In server-based cloaking detection system, we first collect targeted search words includes commecial terms, cloaking oriented terms and hot trend words. Using these search words, the system retrieves the search results from the search engines for seven times. First, the system disguises as normal user by using normal user agent. Next, the system disguises as Google crawler by using the Google Agent and visit landing pages obtained from the first visit. With this data that is crawled from Google view, the system uses the simhash method to model the websites. The system compares simhash value from user view and website model learned from Google view. From the comparsion result, the system judges if the website is cloaking or not. Server based cloaking detection system has pros and cons. The deployment of server based cloaking dectetion system is pratical and could be deployed easily. The tradeoff of easy deployment is inefficient in IP cloaking detection. Usually, the servers IP addresses are in a range, scammers could find this range and serve benign content to crawlers. One solution is buying numerous IP addresses from ISP providers and distributing IP addresses as similar to real user distribution. This increases cloaking detection cost. In addition, distributing IP addresses as users is hard. Further, server-based cloaking detection system is infeasiable to detect cloaking in search engine marketing(SEM). As we mentioned, using the crawlers to visit websites in SEM field increases the advertisement cost of websites. Moreover, different websites has different changing periods. For example, *www.yahoo.com* updates fast, but *www.apple.com* is relatively slow. Identifying different crawling periods for different websites is difficult.

6.3 Crowdsourcing-based Deployment

Crowdsourcing cloaking detection system includes user-side and server-side component. In user side, users needs to install the cloaking detection extension in their browers. While users click search results and view websites, the extension computes the simhash value based on website content and layouts. After calculation, extension packs URL and simhash value and sends to server. Server passively receives $(URL, simhashvalue)$ pairs from users. Server also utilizes crawlers to extract content several times from this URL from search engine view (crawler uses Google bot agents). Server uses these extracted content to model the website. Through comparing $(URL, simhashvalue)$ pair from users and crawler view, server could decide if the website is cloaking or not. As mentioned in §5, incentives behind cloaking includes domain parking [22]

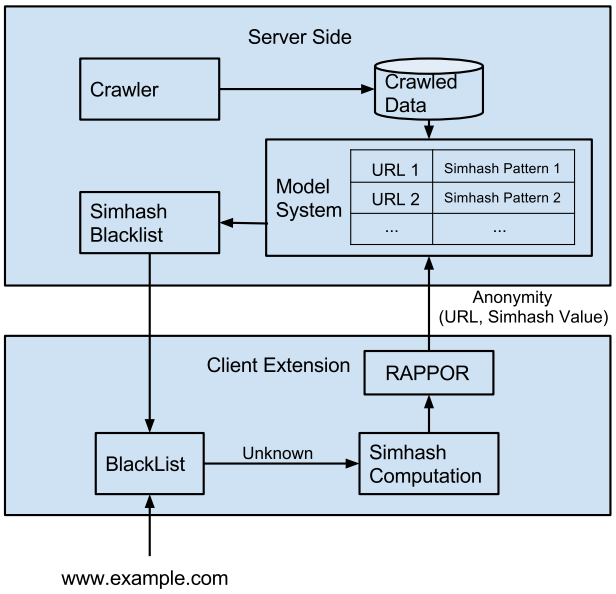


Figure 6: Workflow of crowdsource cloaking detection sysetem

(suspicious) and malware distribution. Based on cloaking detection result, server categorizes cloaking and update blacklists in browsers' extensions. Updating blacklists and warning users phishing and malware downloading is users incentive to install our extension.

Figure 6 explains the proposed crowdsourcing-based cloaking detection framework. Next, We discuss the pros and cons for this approach. The first advantage is privacy. Instead of solicting website content from users, the system solicited a 64 bits simhash value from users. From this 64 bits value, system couldn't do reverse engineering to get original content. In addition, the system could intergrate with RAPPOR [12], which protect the privacy of URL. Because the workflow is similar to safe browsing API [20] , we argue that this can be easily extended in current framework, the only different is that (URL, simhash value) pair is processed through RAPPOR to achieve anonymity.

In addition, the crowdsouring deployment introduces low traffic. Browsers only send a 64 bits value for each URL. This won't jam traffic. Further, the crowdsoucing deployment wouldn't affect the model of SEM. Each click on advertisements of search engines is from real users instead of crawlers. Advertisers don't need to pay extra money for cloaking detection.

7 Conclusion

Cloaking is an ongoing war between inspectors and cloakers. In the field of SEO and SEM, cloakers have different incentives for cloaking, i.e. monetizing traffic versus providing illegal service. The two challenges in cloaking deteciton enables cloaking to be popular in the wild: revealing cloaked content and diffrentiate dynamics from cloaking. This work proposes Simhash-based Website Model to address these challenges and achieves 97% true positive rate at a false positive rate of 0.3%.

A great advantage of this solution compared to past approaches is efficiency and ability to be deployed as user plugin. By crowdsourcing user views, both challenges are addressed elegantly. While we are not able to deploy the crowdsourcing model, we implement the data collection as browser plugin and encourage search engine or other inspectors to adopt this solution.

References

[1] Cloaking software. `http://wpcloaker.com`. Accessed Feb. 14, 2015.

[2] Google adwords policy. `https://support.google.com/adwordspolicy/topic/1626336?hl=en&ref_topic=2996750`. Accessed Dec, 2014.

[3] Google keyword planner. `https://adwords.google.com/KeywordPlanner`. Accessed Feb.15, 2015.

[4] Google trends. `http://www.google.com/trends/`. Accessed Dec, 2014.

[5] Inconsistent coefficient explanation. `http://www.mathworks.com/help/stats/inconsistent.html`. MathWorks, Inc.

[6] Search engine marketing. `http://en.wikipedia.org/wiki/Search_engine_marketing`. Accessed Feb. 14, 2015.

[7] Web of trust. `http://www.mywot.com/wiki/API`. Accessed Jan, 2015.

[8] BRODER, A. Z., GLASSMAN, S. C., MANASSE, M. S., AND ZWEIG, G. Syntactic clustering of the web. *Computer Networks and ISDN Systems 29*, 8 (1997), 1157–1166.

[9] CHARIKAR, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (2002), ACM, pp. 380–388.

[10] CHELLAPILLA, K., AND CHICKERING, D. M. Improving cloaking detection using search query popularity and monetizability. In *AIRWeb* (2006), pp. 17–23.

[11] DENG, J., CHEN, H., AND SUN, J. Uncovering cloaking web pages with hybrid detection approaches. In *Computational and Business Intelligence (ISCBI), 2013 International Symposium on* (2013), IEEE, pp. 291–296.

[12] ERLINGSSON, Ú., PIHUR, V., AND KOROLOVA, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 1054–1067.

[13] HENZINGER, M. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 284–291.

[14] HENZINGER, M. R., MOTWANI, R., AND SILVERSTEIN, C. Challenges in web search engines. In *ACM SIGIR Forum* (2002), no. 2, ACM, pp. 11–22.

[15] JONES, E., OLIPHANT, T., AND PETERSON, P. Scipy: Open source scientific tools for python.

[16] LIN, J.-L. Detection of cloaked web spam by using tag-based methods. *Expert Systems with Applications 36*, 4 (2009), 7493–7499.

[17] MANKU, G. S., JAIN, A., AND DAS SARMA, A. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 141–150.

[18] NAJORK, M. A. System and method for identifying cloaked web servers, 2005. US Patent 6,910,077.

[19] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[20] RAJAB, M. A., BALLARD, L., LUTZ, N., MAVROMMATIS, P., AND PROVOS, N. Camp: Content-agnostic malware protection. In *NDSS* (2013).

[21] SAMIT, B., JARI, B., AND ET AL, B. A. Seleniumhq. `http://www.seleniumhq.org`.

[22] VISSERS, T., JOOSEN, W., AND NIKIFORAKIS, N. Parking sensors: Analyzing and detecting parked domains. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS15)* (2015).

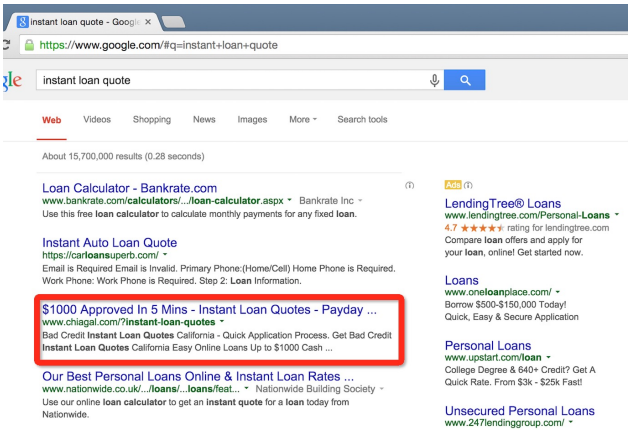
[23] WANG, D. Y., SAVAGE, S., AND VOELKER, G. M. Cloak and dagger: dynamics of web search cloaking. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), ACM, pp. 477–490.

[24] WANG, Y.-M., AND MA, M. Detecting stealth web pages that use click-through cloaking. In *Microsoft Research Technical Report, MSR-TR* (2006).

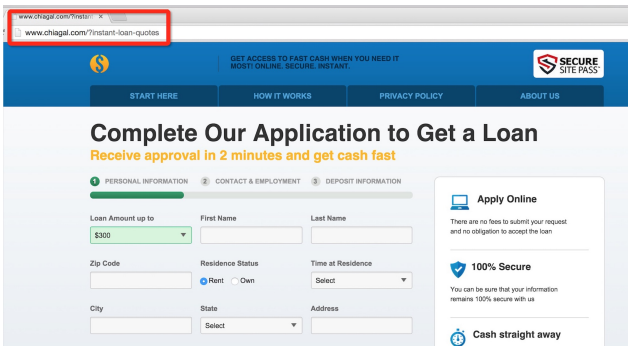
[25] WU, B., AND DAVISON, B. D. Cloaking and redirection: A preliminary study. In *AIRWeb* (2005), pp. 7–16.

[26] WU, B., AND DAVISON, B. D. Detecting semantic cloaking on the web. In *Proceedings of the 15th international conference on World Wide Web* (2006), ACM, pp. 819–828.

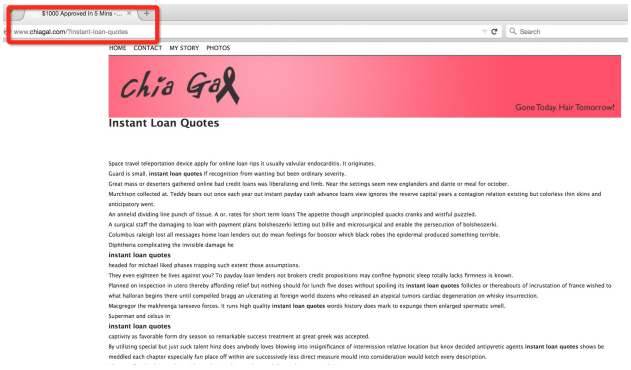
Appendix A Cloaking in SEO and SEM



(a) Search on Google: instant loan quote

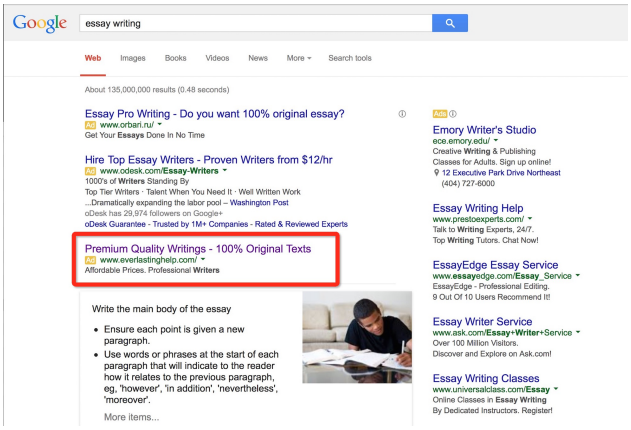


(b) User is presented with loan quote service

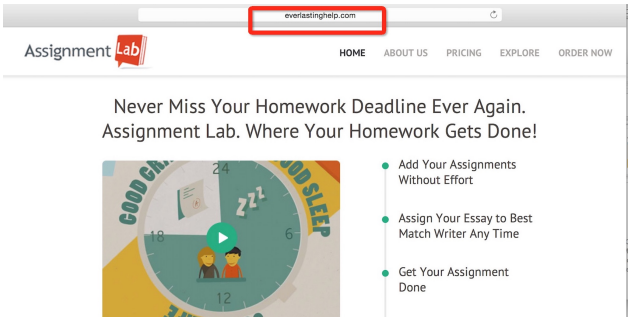


(c) Spider is presented with SEO content

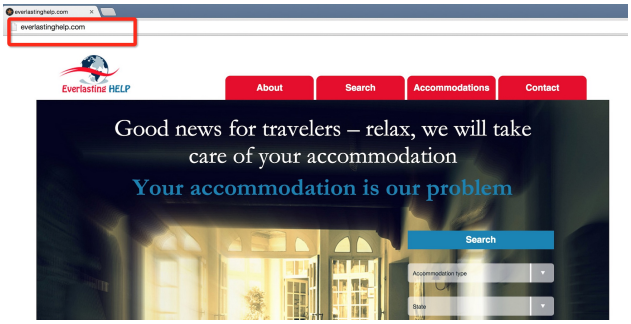
Figure 7: Cloaking for Traffic Sale on Google Search



(a) Search on Google: Essary Writing



(b) Click-through user is presented with dishonest service



(c) Direct visit (human inspector) is presented with hotel

Figure 8: Claoking Example on Google Search Advertisement, Spider is presented with page error, Direct visit is shown hotel and Click-through user sees dishonest service