Website Privacy Preservation for Query Log Publishing

Barbara Poblete
Web Research Group
University Pompeu Fabra
Barcelona, Spain
barbara.poblete@upf.edu

Myra Spiliopoulou
Otto-von-Guericke-University
Magdeburg, Germany
myra@iti.cs.uni-magdeburg.de

Ricardo Baeza-Yates
Yahoo! Research
Barcelona, Spain
ricardo@baeza.cl

ABSTRACT

In this paper we study privacy preservation for the publication of search engine query logs. In particular, we introduce a new privacy concern, which is that of *website privacy* (or business privacy). We define the possible adversaries that could be interested in disclosing website information and the vulnerabilities found in the query log, from which they could benefit. In this work we also detail anonymization techniques to protect website information, and explore the different types of attacks that an adversary could use. We then present a graph-based heuristic to validate the effectiveness of our anonymization method, and perform an experimental evaluation of this approach. Our experimental results show that the query log can be appropriately anonymized against a specific attack for website exposure, by only removing approximately 9% of the total volume of queries and clicked URLs.

General Terms

Algorithms, Security

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous

Keywords

Query Logs, Privacy, Websites

1. INTRODUCTION

Search engine usage logs, specifically *query logs* are extremely rich sources of information, from which the scientific community can benefit immensely. These logs allow, among other things, mining interesting user behavior patterns and rules, that later can result in important innovations, such as improvements to ranking applications, user models, spam detection, and many others. Unfortunately, most companies and owners of query logs do not publish their data, due to important privacy concerns, since it has been demonstrated that even anonymized query logs pose too great a risk of disclosing private user information, when confronted with an adversarial attack.

The concern about query logs and privacy was triggered by the publication in 2006 of the American Online (AOL) query log [1]. This data, which included 20 million Web queries from 650,000 AOL users, underwent only naive anonymization before being made available to millions of people on the Web. The immediate problem with this massive release was that many of the users who were registered on the log had issued personally identifying (or semi-identifying) queries, which combined with other searches, made it possible to map some of them to real people [5]. A few of these people even had their identities published along with their searches [7], which in many cases represented private information. This situation, very critical from the point of view of user privacy, has generated awareness on this topic, and it is doubtful that any other releases of this kind of data will be made until more guaranties of privacy are given.

Privacy preservation in query logs is practically a new scientific matter, that is just beginning to be investigated. Because of this, not even logs that have undergone anonymization are considered safe for publishing. Research still needs to establish all of the possible risks involved in sharing or publishing a query log, to assess the security level of the different anonymization schemas.

Up to now, the only privacy concern that has been mentioned, related to query logs publication, is that of preserving *user privacy*. While many previous anonymization efforts have been aimed in this direction, little attention has been paid to another type of privacy concern, which we consider no less and perhaps even more important: *website privacy* or *business privacy*.

In this work we will argue that important and *sensitive* information about websites can be discovered from query logs, and that naive forms of URL anonymization, such as the one used by AOL, are not sufficient to prevent adversarial attacks. Confidential information about websites, which can be disclosed from query logs, include visits to each document in a site, queries used to reach documents in the site, and keywords that position the business on the Web. Such information is confidential, because websites serve as communication, advertisement and, often, sales channel. Hence,

the traffic recorded in them delivers a picture of customer-company interaction, possibly for the whole product portfolio. One may argue that this recorded traffic is already private to the website owner. However, for major search engines, the traffic they deliver to websites amounts to an important part of the overall traffic to the site. Therefore, if the traffic from search engines is revealed, this will be a very close approximation to the actual *access logs* of a website.

An important issue which makes this problem different from other privacy preservation problems, is that an *adversary* can aggregate public and private data sources when performing an attack. This is especially true when the adversary represents a particular website or company that is interested in finding confidential information about its competitors. In this particular case, the attacker could use their own background knowledge in the area to infer information from the log. This knowledge can include (but is not necessarily limited to) popular queries from which searches reach both competitor sites and the attackers own website query logs; these logs can then be used to decrypt an anonymized log (see Figure 1). Depending on the amount and quality of the information revealed, *industrial espionage* or malicious intent could be argued by the affected parties against the company that provided the log.

Unlike *user* privacy preservation, which is not a well defined concept with respect to who and what exactly needs to be protected, *website* privacy preservation is aimed at protecting a website that represents (or belongs to) a company or other type of organization. In many cases, websites reflect the complete range of services and products offered by those companies. Hence, the traffic recorded for visitors is a faithful image of customer-company interaction. Thus, a thorough Web usage mining analysis may deliver more information than a market study, including insights on the effectiveness of advertising campaigns, popular products, less popular products, number of visits, and how these patterns change over time. In general, any well respected website has some sort of data mining tool to analyze its traffic, and considers methods that will position itself better on Web search engines.

We believe that by anonymizing websites in query logs, user privacy will also be improved, since much sensitive information about users comes from learning which pages they have visited. For example, people may not want to disclose documents they visited related to political or religious issues, health problems, job inquiries, and so on.

The main contributions of this paper are:

- we introduce a new privacy preservation concern in query logs: *website privacy*,
- we show attacks on query logs that disclose website confidential information, and ways to prevent them,
- we create a graph-based method to establish the effectiveness of the main proposed anonymization method, and
- we perform an experimental analysis over real data to validate our heuristic.

Although query log anonymization does not look promising in the near future, especially from the user privacy perspective, we believe that reasonable measures can be taken to preserve website confidentiality. By discussing some of the existing threats and ways to prevent them, we can set a precedent for data mining applications on logs, and future query log publishing, so that the resulting information is inspected to prevent privacy leaks.

2. RELATED WORK

Data mining is the process of automatically (or semi-automatically) discovering knowledge from large amounts of data [16]. This knowledge is generally obtained in the form of interesting patterns and rules. *Web mining* [14], in particular, is the process of applying data mining techniques on Web data, and it can be categorized into three main types: *Web structure mining*, *Web content mining* and *Web usage mining*. Web structure mining focuses on analyzing the hyperlink graph. Web content mining extracts useful information from Web page contents, and Web usage mining mines user access patterns on the Web, which are mostly extracted from Web server access logs, which record all the requests (clicks) made by users [12].

The recent developments in data collection and propagation, along with the rapid growth of the Web, have given rise to a new field of research which is *privacy preserving data mining* [15]. This responds to the fact that nowadays threats against privacy are very common and deserve serious consideration. Privacy preserving data mining aims at analyzing databases and data mining algorithms to study the possible harmful effects that they could have on privacy, and how to prevent them. The main considerations towards preserving privacy are to hide or modify sensitive raw data, such as names, credit card numbers, addresses, and to exclude any type of sensitive knowledge that can be mined from the database, since this is just as compromising. Privacy concerns can be related to information about users, as well as information about companies. It is important to note that many privacy preserving algorithms are based on heuristics. This is because of the premise that selective data modification or sanitization is an NP-Hard problem.

The evaluation of privacy preserving algorithms [15] is usually centered on the following features: the *performance* of the proposed algorithm, the *data utility* after the application of the technique, the *level of uncertainty* with which the sensitive information can be predicted, and *resistance* to different data mining techniques.

General work in privacy preservation on databases discusses how to approach this issue in data publishing [9, 10], and how to keep the utility for data mining purposes. Previous work also studies how to prevent adversarial data mining in relational data bases [3] in the presence of correlation among different fields. Related work also addresses the problem of releasing data in a way that if it is attempted to retrace to whom the data refers, this will lead to at least k entities. This method is called k-anonymity [13]. Although k-anonymity is quite effective, it cannot be directly applied to data that expands across multiple databases, as is the case of website privacy preservation.

Privacy preservation related to Web mining has many implications, the most recent and popular is related to the preservation of user privacy in query logs from search engines. Since the AOL issue in 2006, the protection of user identities when publishing or sharing query logs has become an important goal for Web mining researchers. Two papers in this field [11, 2] are relevant to our work. The authors of [11] propose token-based hashing for query log anonymization. In this approach queries are tokenized and a secure hash function is applied to each token. The authors show how statistical techniques can be used to break this type of anonymization and also, how there is no satisfying framework to provide privacy in query logs. In [2], the authors explain many aspects of the AOL query log problem, and show that traditional privacy preservation techniques cannot be applied in a straightforward way to search log anonymization. Specifically for k-anonymity, they argue that it is too costly to apply to data that is changing rapidly (as query logs do). Then, they present two specific solutions for user anonymization in query logs which attempt to balance log utility for research and privacy.

To the best of our knowledge ours is the first paper to address the issue of privacy preservation of websites or businesses in query logs. It is important to note that even if the problem of preserving user privacy were solved, the solution would not suffice to guarantee website privacy.

3. THE WEBSITE ANONYMIZATION PROBLEM

For our analysis, we assume that any query log has at least the same fields as the one published, and later withdrawn by AOL. Based on this, we define our default query log format as:

{*AnonID, Query, QueryTime, ItemRank, ClickURL*}

Where *AnonID* corresponds to an anonymized user ID, *Query* is the search string, *QueryTime* is the time at which the query was issued, *ItemRank* is the rank of the document clicked as a result of the query, and *ClickURL* is the URL of the document that has been truncated at the website name (e.g. `www.example.org/somepage.html` is reduced to `www.example.org`). It is important to note that even though the AOL log used an anonymized version of the user ID and a truncated version of the clicked URLs, these were easily broken, as detailed in [2]. For our analysis, we also consider the hostname as a central concept and define a *website* as a set of pages under the same hostname.

Anonymizing query logs for data mining applications is especially challenging because *all* attributes in the log are potentially useful, depending on the task being performed. Also, the attributes of the query log are not independent. Using these dependencies, an adversary may deduce the value of a hidden field from an available field. For example, it is well known that queries in search engines show a very strong frequency distribution. This property can be exploited to decrypt anonymized queries from a query log, by using frequency and co-occurrence of terms in a (non-anonymized) reference log [11]. Additionally, query logs have *sequential* records, which means that records cannot be rearranged or shuffled as part of the anonymization process without destroying important temporal and order-dependent information (including user sessions).

Nevertheless, data mining mostly focuses on extracting knowledge, so the exact values of each attribute are not always necessary and can be hidden behind an anonymized value that preserves their "distribution". The difficulty, in this case, relies on deciding which attributes can be generally anonymized or hidden, without limiting mining applications, and at the same time minimizing the amount of private information that an adversary could extract. This is a very complex optimization problem. We approach it in a conservative way by describing possible anonymizations, showing how they could be broken by an adversary, and then increasing the level of hidden information until the log becomes sufficiently resistant to website privacy-disclosing data mining.

Before we discuss further anonymization issues, we define the scope of our work: We describe and focus *only* on the problem of *website privacy preservation* or prevention of *website exposure* (singling-out a website), when publishing or sharing search engine query logs. This means that our goal is to prevent an adversary from obtaining confidential information about traffic to websites, which have been recorded in a search engine's query logs.

In our model for website privacy preservation, we distinguish between two types of adversaries:

1. **General Adversary:** This type of adversary is "just" trying to discover any useful information about as many websites as possible, without any particular site in mind. This type of adversary might show up as a search engine optimization company or other institution that performs market studies.
2. **Particular Competitor:** This type of adversary is a website or company that tries to disclose information about its competitors from the query log. In many cases, such an actor has already some information about the market share, portfolio and activities of the competitors, and can impute this background knowledge upon the anonymized log to de-anonymize it. One of the most important pieces of data that this actor can exploit is its own query log, which can be used to recreate pieces of the anonymized query log.

In query log privacy preservation, we must also take into consideration the different data sources that can be combined to assess additional information (see Figure 1). Some of these sources may be publicly available, while others may be private. They include:

1. an *anonymized search engine query log* (public),
2. actual *search results for queries* (public), from a search engine, maybe even the same search engine from where the log came from, or another one with similar document coverage, and
3. *access logs from a given site* (private), which contain clicks to the site from external search engines. This

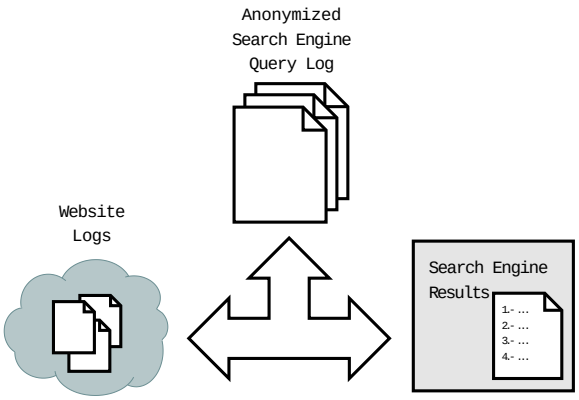

Figure 1: Different data sources involved query log privacy preservation.

information is quite easy to retrieve, from the *referer*[1] field in a standard combined access log. A particular competitor will possess such a log for the own website.

The combination of sources for data mining can be viewed as data mining over multiple databases. Presently, this problem is not covered by current k-anonymity solutions.

Example: To illustrate the challenges of combining sites for data mining, let's consider an on-line computer hardware store "Site A". Site A knows that the most popular queries used to reach it are: *refurbished computer*, *cheap notebook*, *laptop*, *memory* and *desktop computer*. Site A has access to a published search engine query log (such as the AOL log), and it wants to discover as much information as possible about its competitors. It first assumes that its competitors are reached via the same queries, which form an initial list of keywords L_0. This initial list can be expanded by searching for the URLs of Site A's most important competitor "Site B" in the query log. This allows Site A to obtain additional keywords from Site B, such as *electronics store*, *portable computers* and *computer deals*. The result is an enhanced list of keyword terms L_1.

Next, Site A issues the queries in L_1 to an on-line public search engine and discovers which *other sites* are returned in each result set. Site A discovers that its competitors are next to Site B, also Site C and Site D. Furthermore, now that site A knows who its competitors are, it can extract all of the usage information for each one of those sites from the published query log, discovering which site has more visitors, more hits from the search engine, which queries reach other sites but do not reach Site A, and so on. As a consequence of knowing this information, Site A can choose to advertise exactly on the keywords that are the most popular for reaching sites B, C and D, or it may focus all of its marketing resources on the weakest competitor. Most importantly, Site A can make business decisions based on information that was not available before the disclosure of the log. □

[1]This is a misspelling of "referrer" and is the term officially used in the HTTP specifications.

4. ATTACKS AND MEASURES AGAINST WEBSITE DISCLOSURE

In this section we will discuss an incremental approach to anonymize a query log. In each step we will show how an adversary could go about doing an attack to discover information about a certain site, or other site related information (i.e., who are its competitors). We will undertake this task by establishing our analysis on three main types of attacks, which we define as: *attacks on vulnerable queries*, *attacks using a website log* and *attacks with user information*. It is important to note that although we cannot attempt to find *all* possible weak points in a log, our aim is to establish that several weak points exist and that different techniques can be used to prevent them.

4.1 Attacks on Vulnerable Queries

As explained before, the anonymization used in the AOL log was not sufficient for user and website privacy preservation. Although the clicked URLs are truncated at the site level, the log still provides the original *Query* and *ItemRank* information, which is enough to do a look-up on the AOL search engine and discover most of the actual URLs for each *ClickURL* field. As a result, all site information available on the original query log is disclosed.

From the beginning it is necessary to increase the restrictions to the log data. A simplistic approach to prevent the previous attack would be to hide the *Rank* attribute in the log and at the same time do a simple anonymization on the *ClickURL*. The idea here is to replace this attribute with a unique identifier. This can be done using three methods:

1. By assigning a unique ID to every URL (this loses information about which URLs are in the same domain).
2. By assigning unique IDs to each URL and using IDs that identify Web pages that belong inside a same site or domain, e.g. using correlative ID numbers for a same site (this gives more information than method 1, such as number of clicked pages for a site, in which occasions different pages the same site are clicked, etc).
3. Assign a unique ID to all URLs in a website. This still allows an analysis of the click distribution and other useful statistics for rank experiments, but it does not allow documents inside a website to be distinguished.

Let us assume that we choose to anonymize URLs using method number 2, which allows us to preserve the most information in the log about general website properties.

We will consider the case in which an adversary has only access to public data sources, such as the anonymized query log and a public search engine. This would be the case for most "general adversaries". In this scenario an intuitive way to obtain information about any website is by exploiting certain types of *vulnerable* queries, such as some *navigational queries* [8]. These are queries where the user knows exactly the page that they are trying to reach and they use the search engine to obtain the URL, using it like a bookmark on a browser. Navigational queries become "vulnerable" from a privacy preservation point of view when

they include only the terms that later appear in the root of the selected URL (i.e. the website root). For example queries such as "amazon" for which the user clicks on the link `http://www.amazon.com`. From these queries, even in a log anonymized as ours, it would be possible to discover the actual website to which they belong to. To prevent this, the query log should be checked for queries that meet this criteria (keywords that match the URL root string) and, either remove them, or hide the search terms.

Another type of "vulnerable queries" are those that return fewer than k results (for k-anonymity) from the search engine. If the result set of that query is equal to k (for k small enough, i.e. equal to 1) then the URL ID in that result set can be successfully mapped to a real URL. To prevent this situation, the anonymized log should have removed all queries whose result sets are smaller than k.

The last and more complex type of attack that we will mention, under the category of "vulnerable queries", is an attack that occurs when we have pairs of queries in the query log that have intersections between their clicked result sets. We will start by assuming that the adversary is a "particular competitor" that is trying to find information about other specific sites, and later we will generalize this to any type of attacker. Given this situation, the following heuristic can be used to break the anonymization:

1. The adversary defines a set of queries Q_1, for which it knows that the URLs of the websites that it considers as its competitors will show up on the highly ranked search results of the search engine.
2. The adversary does a look-up of the occurrences of the queries in Q_1 in the anonymized query log L, from which it obtains a set of anonymized URLs that represent the "candidate competitors" set CC. The task at hand is to map as many URLs $u \in CC$ as possible, to their real URLs, so once their identities are established, all the information about the relevant sites can be extracted from L.
3. For each anonymized URL $u \in CC$, the adversary collects *all* of the unique queries q that have u as a clicked URL in L, we will call this set Q.
4. Then, for each $q_i \in Q$ the adversary collects its anonymized result set, which we will call R_{Ai}.
5. Then, for each query that belongs to a pair $(q_i, q_j) \in Q$, for which $|R_{Ai} \cap R_{Aj}| \geq 1$, we issue it live to the search engine an recover its result sets R_i and R_j. If $|R_{Ai} \cap R_{Aj}| = |R_i \cap R_j| > 1$, then it is known that the URLs in $|R_{Ai} \cap R_{Aj}|$ have been *approximately* mapped to real URLs. This becomes an *exact* match if $|R_{Ai} \cap R_{Aj}| = 1$, or if all URLs in $|R_{Ai} \cap R_{Aj}|$, except for one, have already been discovered using the same methodology.

This attack can be extended for a *general adversary* by searching the complete log for any intersections among two clicked result sets, and adding the queries that generated them to Q, and then applying steps 4 and 5 from the previous heuristic.

This type of attack does not guarantee that an adversary will find all the real URLs it wants to, but it could potentially disclose some site URLs, which could be used to disclose *all* of the URLs from the websites that they belong to. We believe that this attack is very interesting and also the one that is more likely to disclose website information. To prevent this vulnerability, we propose to remove from the log any one of two queries that share at least one clicked result (or URL ID). A detailed heuristic to do this on a query log will be presented in Section 5.

4.2 Attacks Using a Website Log

At this point, we have reviewed the case where an adversary has access only to the anonymized log and to the results of a public search engine. Now we will look into the case in which the adversary is a particular site and has access to its website's access log.

Website access logs register all of the requests made to the website, including URLs, time of the request, User Agent, IP address, and in the case of combined logs, they record the originating URL of the request (or referrer URL, from which the link to the current page was selected). If the request was generated from the results page of a search engine, this will be logged into the website's access logs, and the keywords and URL of the search engine can be extracted when processing this log.

In this context, if an adversary has an access log for the same period of time in which the anonymized log was generated, they could use this information to attempt to break the anonymization of the query log. For instance, they can find what URL ID was assigned to them in the anonymization process. This alone is not a problem in most cases. The problem arises if more sites colluded and shared their logs. The combination of these logs would enable them to discover many URL IDs using the previous attack.

This is not the most probable situation, but it cannot be discarded completely, and it could be prevented by adding an extra restriction on the result sets of query that has an intersection with another result set: *that they have to contain URLs that belong to more than k different sites*, and fix k at an acceptable level. Since we consider this case less likely than the others, we will not go more in-depth into its analysis at this time.

4.3 Attacks with User Information

Another more specific attack is the case of the adversary identifying a single user on the log, meaning that for a certain user, the adversary would know what its clicked results were. This can be a threat because the real URLs the user clicked on are known to the attacker and if that user is discovered in the anonymized log, they can map those URL IDs. If for example the intention is to have a periodical release of query logs, the adversary could have a particular user submit queries to the search engine that it knows are going to bring up results of its competitors and use them to later map them in the log.

This issue is avoided by preventing the identification of a single user in the query log, which is addressed as a separate

problem (see "Related Work" Section) and is beyond the scope of this paper.

5. GRAPH-BASED METHOD

In Section 4.1 we presented an attack on query logs that relies on the occurrence of intersections among result sets of queries. As a way to overcome this particular weakness, these intersections need to be removed from the log completely, or their number should be very low. This would guarantee a certain level of security in website anonymization regarding this particular attack.

To analyze the vulnerability of query logs to this attack and at the same time to see how to reduce the number of intersections between results sets of queries, we have designed a graph-based methodology. This graph representation of the query log consists of representing queries as nodes, which are joined by an undirected edge if they share at least one URL between their clicked results set. Using this graph approach we show that the solution to the attack described in Section 4.1, is a *well defined optimization problem*, that consists of disconnecting the graph by removing nodes and at the same time preserving as many nodes as possible. This type of problem is also known as finding the *maximum independent set*, and is NP-Hard. Due to the difficulty involved in disconnecting the query log graph, we define a heuristic approach.

The first step is to define a general measure of graph *density* this can be represented by a measure of how likely it is to find an edge among any two nodes in the query log. We define this with the formula:

$$\text{Density} = \frac{2(\#\text{ edges})}{\#\text{ nodes }(\#\text{ nodes-1})}$$

Thus, the goal of our heuristic approach is to have Density equal to 0, which means that there are no intersections among result sets in the log.

One indication that the number of edges in the graph can be quickly reduced by removing nodes, is if the distribution of number of edges per node follow a power law. This would indicate that by only removing a few high-degree nodes in the graph, it would become disconnected very fast [4]. The next step is to measure how many nodes must be removed to completely disconnect the graph. For this we define the following heuristic:

1. Sort nodes by their degree.
2. Compute the value of Density.
3. Remove the node with the highest degree.
4. Recalculate the degree of all the neighbor nodes to the node that was removed.
5. If the value of Density is *not* equal to 0, then go to step 1, otherwise finish.

Once the graph has been completely disconnected, certain characteristics can be analyzed, such as the speed at which the value of Density decreases and how much of the log's contents have to be removed to completely disconnect the graph.

6. RESULTS AND DISCUSSION

In this section we present an experimental evaluation of the methodology described in Section 5 on real data. For this we use log files from the Yahoo! search engine. For privacy reasons these logs are carefully controlled and cannot be released for general study. Even for this analysis we do not deal with the raw log data, but only with its graph representation. The graph representation is an application of the graph models developed in [6], and can be computed rapidly. This took approximately 2 hours on a dual core AMD Opteron(tm) Processor 270 with 6.7 gigabytes of RAM (although the processing always used less than 4 gigabytes of memory and only one CPU). The log used for this experiment contains about 3 million nodes and reflects a sample of the usage registered by the search engine in 2005.

From the beginning, the query graph has a Density value equal to 0.000089, which can already be considered low (the highest possible value is Density = 1).

First, we computed the likelihood of finding intersections (edges) among queries that share a term, because this could be a possible way of starting an attack on the query log graph. Although the Density in this case is also low, equal to 0.000045, it is not lower than the general Density for the whole graph. This means that queries that share terms do not necessarily share more clicked results, and this would not be a better way for an adversarial to search for edges in the graph.

In a worst case scenario we can assume that an attack can start in a well connected component of the graph, where the chance of finding edges would be stronger. Thus, we focus on analyzing only the largest connected component of our graph data. For this we extract all of the connected components on the graph (see size distribution in Figure 2). We found that there is a very big connected component which includes almost 50% of all of the nodes in the graph. Without loss of generality, we study this big connected component, and analyze the distribution of the node degrees. If this distribution corresponds to a *power law*, then, by removing the nodes with the highest degrees it would be possible to disconnect the graph very quickly. This would show that this query log is very likely to be successfully anonymized with very little loss of information. As Figure 3 shows, the degree distribution is *not* a power law, as it seems that there are at least about 9% of the nodes with very high degree. This does not demonstrate that by removing them the graph will become disconnected.

The next step to attempt to disconnect the graph is to proceed with the heuristic described in Section 5. We continue by measuring how the likelihood of finding an edge between any two nodes in the graph decreases, when removing the nodes with the highest degree. The graph for this experiment is presented in Figure 4. From this analysis we find that after removing 4.64% of the nodes in the graph it be-

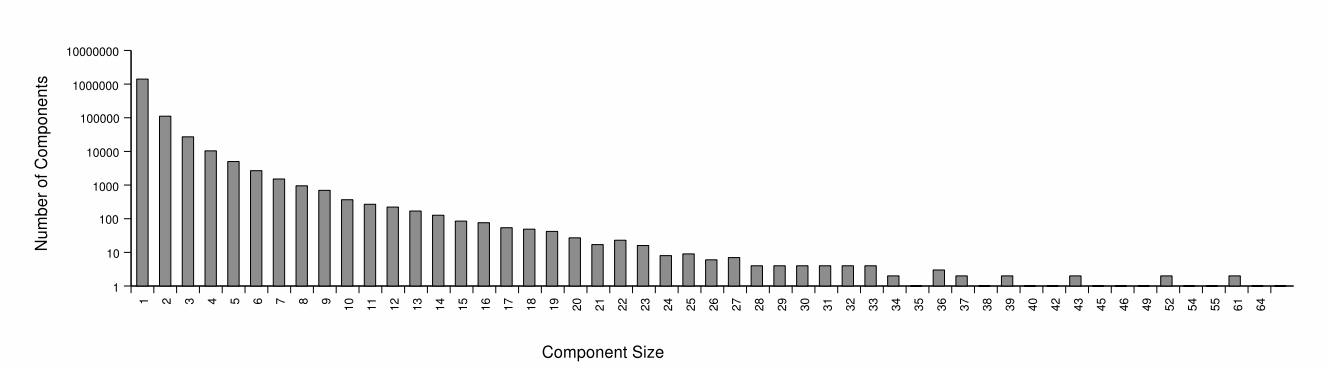

Figure 2: Component size distribution in the query log sample.

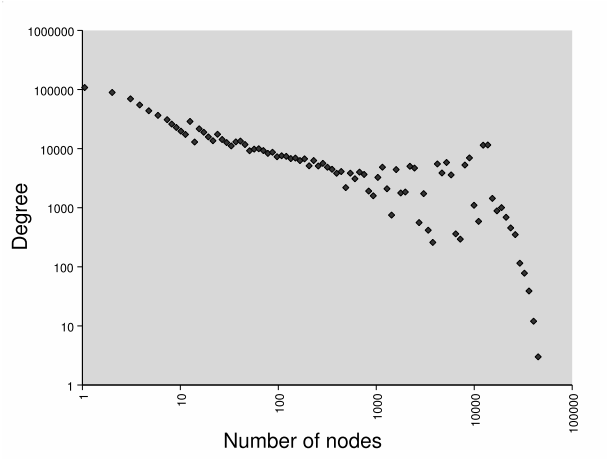

Figure 3: Degree distribution in the query log sample.

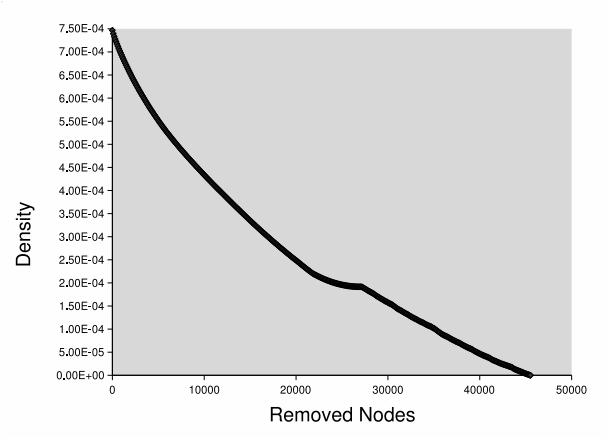

Figure 4: Density Vs. number of removed nodes.

comes disconnected, and that these nodes represent 9.5% of the total number of queries found in the log, and 9.2% of the total number of clicked documents on the log. These values are very low and show that the query log can be anonymized without sacrificing too much of its contents. The queries that were removed are not very frequent which implies that in general they are less useful for data mining purposes. This produces only a small loss of potentially interesting information in the log. Although, it is important to mention that there always has to be a trade-off between the amount of log preserved in the anonymization, and the strength that it has for privacy preservation.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a new issue, which is preserving website privacy. We have shown how website information can be extracted from naively anonymized query logs, including a quite complex attack, and techniques to overcome these weaknesses. We have introduced a graph-based representation for query log privacy preservation analysis. We have also defined different types of adversaries encountered when dealing with website information, as well as general types of vulnerabilities, which can be used to disclose information.

Our experimental results on real data show that it is possible to anonymize a query log for a specific and complex type of website attack. This can be achieved by removing a small fraction of the volume of the actual log data (9.5% of the queries and 9.2% of the clicked documents). Also, graph statistics obtained from research related to [6] and the usually stable distributions found in query logs, show that it is very likely that results obtained from this log can scale to logs of more extended periods of time, and to query logs from other Web search engines. Since experimental results show that only a reduced portion of the log needs to be removed, this should allow almost all data mining tasks that do not require knowledge about a specific website to be performed. Also, the queries that are removed are infrequent, minimizing the loss of potentially useful information in the remaining data.

Another important characteristic of the heuristic presented in this work is that the graph representation of the query log can be computed relatively fast. This makes the our anonymization approach suitable for rapidly changing data, such as query logs.

Future work on this topic includes exploring different types of attacks for website disclosure, as well as studying other business privacy issues on logs. One possibility is the analysis of IP masks on logs, to determine information about user behavior from certain company, which can lead to business privacy disclosure. This can become a problem, for example, if employers from a certain vendor where found to buy or use products from the competition.

8. ACKNOWLEDGMENTS

The authors thank Alessandro Tiberi for providing the graph representation of the query log, along with his help to understand this data. Also we thank Aristides Gionis for many valuable discussions and feedback, and Vanessa Murdock and Bo Pang for their corrections for this final version.

9. REFERENCES

[1] AOL research website, no longer online. http://research.aol.com.

[2] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Query Log Analysis: Social and Technological Challenges, Workshop in WWW '07*, 2007.

[3] C. Aggarwal, J. Pei, and B. Zhang. On privacy preservation against adversarial data mining. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 510–516, 2006.

[4] R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, July 2000.

[5] M. Arrington. AOL proudly releases massive amounts of private data. http://www.techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/, 2006.

[6] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *To appear in ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007.

[7] M. Barbaro and T. Zeller. A face is exposed for AOL searcher no. 4417749. New York Times, 2006.

[8] A. Broder. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3–10, 2002.

[9] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. *Theory of Cryptography Conference*, pages 363–385, 2005.

[10] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 217–228, 2006.

[11] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On anonymizing query logs via token-based hashing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 629–638, New York, NY, USA, 2007. ACM Press.

[12] B. Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer, 2007.

[13] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, 1998.

[14] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[15] V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[16] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.