ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-3, 492 – 497

PPDP-MLT: K−ANONYMITY PRIVACY PRESERVATION FOR PUBLISHING SEARCH ENGINE LOGS

B.Lavanya[1], K. Rajani Devi[2]

[1]M.Tech (IT), Nalanda institute of engineering and technology (NIET), Sattenapalli (M.D.), Guntur(D.T),India, **bathulalavanya@gmail.com**

[2]Assoc.professor (HOD), Dept of IT, Nalanda institute of engineering and technology (NIET), Sattenapalli(M.D.), Guntur(D.T),India, **rajanidevik@gmail.com**

Abstract

In this paper we investigate the problem of protecting privacy for publishing search engine logs. Search engines play a crucial role in the navigation through the vastness of the Web. Privacy-preserving data publishing (PPDP) provides methods and tools for publishing useful information while preserving data privacy. Recently, PPDP has received considerable attention in research communities, and many approaches have been proposed for different data publishing scenarios. In this paper we study privacy preservation for the publication of search engine query logs. Consider an issue that even after removing all personal characteristics of the searcher, which can serve as links to his identity, the publication of such data, is still subject to privacy attacks from adversaries who have partial knowledge about the set. Our experimental results show that the query log can be appropriately anonymized against the specific attack, while retaining a significant volume of useful data. In this paper we study about problem in search logs and why the log is not secure and how to make log secure using data mining algorithm and techniques like Generalization, Suppression and Quasi identifier.

Index Terms: *Search engine, log, Algorithm, Data mining technique - Suppression, Generalization, Quasi identifier*

----------***----------

1. INTRODUCTION:

Have you ever wondered what happens when you type your query into the Google search box and what data we store about that search? [1] Search involves interactions between two parties, a user (*U*) and a search engine (*S*). There are two basic interaction cycles between a user and a search engine: (1) Search: A user *U* composes and submits a query *q* to search engine *S*, and the search engine

S would return some search results *R = fR1; :::;Rng* to the user. (2) Browse: A user *U* chooses to view a Result *Ri 2 R*, and the search engine would bring the user the content of *Ri*. In a search process involving many such interaction cycles, a user thus potentially reveals the following. Three kinds of personal information:

1. User identity: This could be a personal user ID in the case when the user has to register an account, or the IP address of the machine that the user is using.
2. Queries: This includes all the queries the user has submitted to the search engine.
3. Viewed results: This includes all the viewed web pages by the user.

Actually, the user also reveals some context information such as the time stamp. Since such personal information can potentially reveal a gamut of user's private life such as political inclination, family life, and hobbies, disclosing such information, especially in an aggregated fashion, would clearly raise serious concerns for users. One may notice that there is a remarkable difference between user's queries and clicked search results. Since queries are composed by users themselves, thus directly reveal the user's information need, while the search results are composed by the Web page publishers. Thus in general, queries may contain much more personally identifiable information (PII) than viewed search results. However, from the viewpoint of privacy.

Protection, both queries and viewed results can cause concerns for users and the difference appears to be not crucial. Let's take a simple search like "cars."

When someone types the word "cars" into the Google search engine, the request gets sent from that user's computer over the internet to our computers, which look for the right search results. Once our computers have found the results, they send these back to the User's computer, all in a fraction of a second. We then store some data about this exchange: the search query ("cars"), the time and date it was typed, the IP address and cookie of the computer it was entered from, and its browser type and operating system. We refer to these records as our search logs, and most websites store records of visits to their site in a similar way. Here's what a typical log entry at Google looks like:

123.45.67.89 - 25/Aug/2011 10:15:32 -
http://www.google.com/search?q=cars -
Chrome 2.0.0.7; Windows NT 5.1 -
740674ce2123e969

1.1. IP address:

123.45.67.89 is the IP address assigned to the user's computer by his or her service provider. Just like other websites, when you ask Google for a page (a search results page, for example), we use your computer's IP address to ensure that we get the right results back to the right computer. It's important to remember that IP addresses don't say exactly where an individual user is, or who they are. In fact, some Internet Service Providers (ISPs) give users a different IP address every time they log onto the web. The most Google can tell about a user from his computer's IP address is that user's general location (for example, Boston) and possibly the ISP they use to connect to the Internet. Only the ISP (who actually controls the user's account) can match an individual with an IP address.

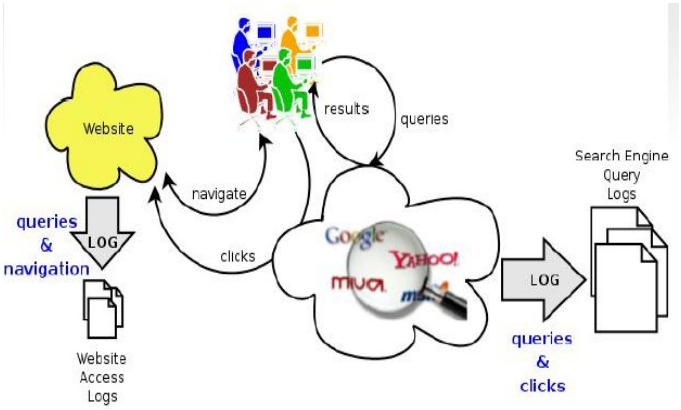
Time and date: 25/Aug/2011 10:15:32 is the date and time the user typed the query into Google.

Search query: http://www.google.com/search?q=cars is the search query, in this case "cars."

Browsers and operating Systems: Chrome 2.0.0.7; Windows NT 5.1 is the browser and operating system being used.

Cookie: 740674ce2123a969 is the unique cookie ID assigned to a browser the first time a user visits Google. Like an IP address, a cookie doesn't tell Google who a user actually is or where they live – it only identifies a computer. You can delete these cookies at any time in your computer's browser.

1.2. ARCHITECTURE:



Internal Architecture [5]: The below figure provides a detailed view of the Search service internal architecture. Following are the components of the Search service's architecture.

Index Engine: Processes the chunks of text and properties filtered from content sources, storing them in the content index and property store.

Query Engine: Executes keyword and SQL syntax queries against the content index and search configuration data.

Protocol Handlers: Opens content sources in their native protocols and exposes documents and other items to be filtered.

Filters: Opens documents and other content source items in their native formats and filters into chunks of text and properties.

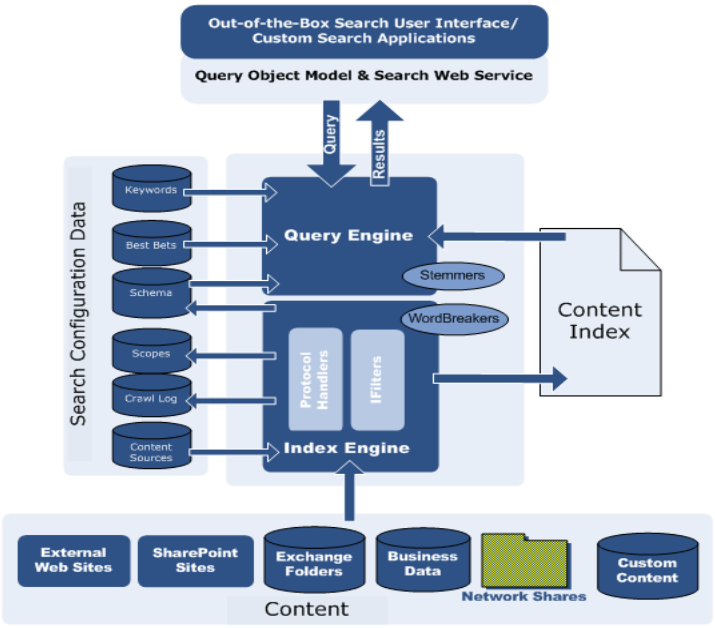
Content Index: Stores information about words and their location in a content item.

Property Store: Stores a table of properties and associated values.

Search Configuration Data: Stores information used by the Search service, including crawl configuration, property schema, scopes, and so on.

Word breakers: Used by the query and index engines to break compound words and phrases into individual words or tokens.

Search Query Execution: When a search [6] query is executed, the query engine passes the query through a language-specific word breaker. If there is no word breaker for the query language, the neutral word breaker is used, which does whitespace-style word breaking, which means that the word breaking occurs where there are whitespaces in the words and phrases.



After word breaking, the resulting words are passed through a stemmer to generate language-specific inflected forms of a given word. The use of word breaker and stemmer in both the crawling and query processes enhances the effectiveness of search because more relevant alternatives to a user's query phrasing are generated. When the query engine executes a property value query, the index is checked first to get a list of possible matches. The properties for the matching documents are loaded from the property store, and the properties in the query are checked again to ensure that there was a match. The result of the query is a list of all matching results, ordered according to their relevance to the query words. If the user does not have permission to a matching document, the query engine filters that document out of the list that is returned.

Logging

Query Log

The information tracked in the query log includes:

1. The query terms being used
2. If Search results were returned for search queries
3. Pages that were viewed from search results

2. PROBLEM STATEMENT

Existing work on publishing logs make Scientists all around the world to tap this gold mine for their own research. The log contains sensitive information and Non-personal information

2.1. Sensitive information

"Sensitive personal information" includes information we know to be related to confidential medical information, racial or ethnic origins, political or religious beliefs or sexuality and tied to personal information.

2.2. Non-personal information

"Non-personal information" is information that is recorded about users so that it no longer reflects or references an individually identifiable user. Thus in any search activity, the information a user U potentially reveals when attempting to satisfy an information need N can be represented as ($ID(U)$; $TEXT(N)$), where $ID(U)$ is some ID revealed about the identity of the user (e.g., a user ID or an IP address), and $TEXT(N)$ is a text description of the information need N (e.g., a set of related queries and/or viewed results). When a user conducts a series of k search activities, the sensitive personal information that the user may reveal can be represented as $P(U) = f(ID(U; i); TEXT(N; i))g$ where $i = 1; :::; k$. The privacy concern of a user is that all or some of the information in $P(U)$ may be captured by some other people in the world. The concern may be less if $P(U)$ is revealed to some "trustable" party (e.g., a search engine company that has a clearly written policy on privacy protection) than to some "untrustable" parties (e.g., any third party who has access to the web search log). Note that $P(U)$ is precisely what is needed to help a search engine better understand the user's information need. Thus performing personalized search in some sense "requires" a user to release $P(U)$. Such tension has created a barrier for deploying personalized search applications, and the main challenge of privacy-preservation personalized search is to exploit $P(U)$ to help improve the search service for U while protecting $P(U)$ as much as we can from being known by anyone else in the world. *The User identity* $ID(U)$ can generally be mapped to a single or a small group of users (e.g., family members) with the help of public databases.

For example, given an IP address, geographic information such as city and state can be known through the *who is* service. This approach introduces uncertainty about individual values before data is published or released to third parties for Data mining purposes. To avoid such existing problems we introduce Apriori based suppression algorithm.

3. IMPLEMENTATION

In this paper our study is based on Apriori based data suppression algorithm. Apriori algorithm used to find relevant search details of users from search engines (ex: Bing, Google, Youtube…). Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation*), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. The purpose of the Apriori Algorithm is to find associations between different sets of data. It is sometimes referred to as "Market Basket Analysis". Each set of data has a number of items and is called a transaction. The output of Apriori is sets of rules that tell us how often items are contained in sets of data.

3.1. Algorithm Pseudocode

The pseudocode for the algorithm [7] is given below for a transaction database *T*, and a support threshold of ϵ. Usual set theoretic notation is employed, though note that *T* is a multiset. C_k is the candidate set for level *k*. Generate() algorithm is assumed to generate the candidate sets from the large item sets of the preceding level, heeding the downward closure lemma. *Count[c]* accesses a field of the data structure that represents candidate set *c*, which is initially assumed to be zero. Many details are omitted below, usually the most important part of the implementation is the data structure used for storing the candidate sets, and counting their frequencies.

Apriori(T, ϵ)

$L_1 \leftarrow \{$ large 1-itemsets $\}$

$k \leftarrow 2$

while $L_{k-1} \neq \emptyset$

$C_k \leftarrow$ Generate($L_k - 1$)

For transactions $t \in T$

$C_t \leftarrow \{c | c \in C_k \wedge c \subseteq t\}$

For candidates $c \in C_t$

$count[c] \leftarrow count[c] + 1$

$L_k \leftarrow \{c \in C_k | \ count[c] \geq \epsilon\}$

$k \leftarrow k + 1$

$\bigcup L_k$

Return k

3.2. Example:

A large supermarket tracks sales data by stock-keeping unit (SKU) for each item, and thus is able to know what items are typically purchased together. Apriori is a moderately efficient way to build a list of frequent purchased item pairs from this data. Let the database of transactions consist of the sets {1,2,3,4}, {1,2}, {2,3,4}, {2,3}, {1,2,4}, {3,4}, and {2,4}. Each number corresponds to a product such as "butter" or "bread". The first step of Apriori is to count up the frequencies, called the supports, of each member item separately: This table explains the working of apriori algorithm.

Item	Support
1	3
2	6
3	4
4	5

We can define a minimum support level to qualify as "frequent," which depends on the context. For this case, let min support = 3. Therefore, all are frequent. The next step is to generate a list of all 2-pairs of the frequent items. Had any of the above items not been frequent, they wouldn't have been included as a possible member of possible 2-item pairs. In this way, Apriori *prunes* the tree of all possible sets. In next step we again select only these items (now 2-pairs are items) which are frequent:

Item	Support
{1,2}	3
{2,3}	3
{2,4}	4
{3,4}	3

And generate a list of all 3-triples of the frequent items (by connecting frequent pairs with frequent single items). In the example, there are no frequent 3-triples. Most common 3-triples are {1, 2, 4} and {2, 3, 4}, but their support is equal to 2 which is smaller than our min support. We consider the following privacy problem: A data holder wants to release a version of data for building classification models, but wants to protect against linking the released data to an external source for inferring sensitive information. We adapt an iterative bottom-up generalization from data mining to generalize the data. This approach incorporates partially the requirement of a targeted data mining task into the process of masking data so that essential structure is preserved in the masked data. The idea is simple but novel: we explore the data generalization concept from data mining as a way to hide detailed information, rather than discover trends and patterns. Once the data is masked, standard data mining techniques can be applied without modification.

Our work demonstrated another positive use of data mining technology: not only can it discover useful patterns, but also mask private information. Generalization has several advantages. First, it preserves the "truthfulness" of information, making the released data meaningful at the record level. This feature is desirable in exploratory and visual data mining where decisions often are made based on examining records. In contrast, randomized data are useful only at the aggregated level such as average and frequency. Second, preferences can be incorporated through the taxonomical hierarchies and the data recipient can be told what was done to the data so that the result can be properly interpreted.

3.3. Suppression:

We consider this method to suppress the data by doing so we can secure the data. The most common method of preventing the identification of specific individuals in tabular data is through cell suppression. This means not providing counts in individual cells where doing so would potentially allow identification of a specific person to be secure.

We extend our work on micro data suppression

(1)To prevent not only probabilistic but also decision tree classification based inference

(2) To handle not only single but also multiple confidential data value suppression to reduce the side-effects.

3.4. Generalization:

A generalization, written fcg(feature coupling generalization) ! p, replaces all child values fcg with the parent value p. A generalization is valid if all values below c have been generalized to c. A vid is generalized by fcg ! p if the vid contains some value in fcg. In the existing paper this technique is used to hide the actual count of the url in the database
(Anonymity for Classification)

Given a relation R, an anonymity requirement < VID; K >, and a hierarchy for each attribute in VID, generalize R, by a sequence of generalizations, to satisfy the requirement and contain as much information as possible for classification. The anonymity requirement can be satisfied in more than one way of generalizing R, and some lose more information than others with regard to classification. One question is how to select a sequence of generalizations so that information loss is minimized. Another question is how to find this sequence of generalizations efficiently for a large data set. In this study, we observed that the generalization novelty factor in that it increased the number of distinct vids faster. When the number of distinct vids is large, the effectiveness of the suppression in "generalized-based" became more significant.

4. CONCLUSION:

We have investigated data mining as a technique for masking data, called data mining based privacy protection. The idea is to explore the data generalization concept from data mining as a way to hide detailed information, rather than discover trends and patterns. Once the data is masked, standard data mining techniques can be applied without modification. Our work demonstrated another positive use of the data mining technology: not only can it discover useful patterns, but also mask private information. In particular, we presented a bottom-up generalization for transforming specific data to less specific but semantically consistent data for privacy protection. We focused on two key issues, privacy and scalability. The scalability issue was addressed by a novel data structure for focusing on good generalizations. The proposed approach achieved a similar quality but much better scalability compared to existing solutions.

5. FUTURE WORK

As with most existing work on perturbation based PPDM, our work is limited in the sense that it considers only linear attacks. More powerful adversaries may apply nonlinear techniques to derive original data and recover more information. Studying the MLT-PPDM problem under this adversarial model is an interesting future direction.

REFERENCES

[1] Michaela G¨otz, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke
Publishing Search Logs – A Comparative Study of Privacy Guarantees – IEEE Transaction on knowledge and data engineering

[2] Alberto Trombetta, Wei Jiang, Elisa Bertino
and Lorenzo Bossi
Privacy-preserving Updates to Anonymous and Confidential Databases – IEEE Transaction on knowledge and data engineering

[3] Privacy-preserving Mining of Association Rules from Outsourced Transaction Databases

[4] Privacy Protection in Personalized Search Xuehua Shen, Bin Tan, ChengXiang Zhai
Department of Computer Science University of Illinois at Urbana-Champaign

[5] Enterprise Search Architecture
http://msdn.microsoft.com/enus/library/ms570748(v=office.12).aspx

[6] Using Search Engines - A Tutorial
http://www.learnwebskills.com/search/engines.html

[7] Apriori algorithm - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Apriori_algorithm

[8] Website privacy preservation for Query
Log publishing – Barbara poblete, Myra
Spiliopoulou and Ricardo Baeza – Yates

[9] http://www.ijcaonline.org/winbis/number1/SPE194T.pdf

BIOGRAPHIES

B LAVANYA, M.Tech (IT),
NALANDA INSTITUTE OF ENGINEERING AND TECHNOLOGY (NIET),
SATTENAPALLI (M.D.),
GUNTUR (D.T),

K.RAJANI DEVI
ASSOCIATE.PROFESSOR, H.O.D.,
DEPARTMENT OF IT,
NALANDA INSTITUTE OF ENGINEERING AND TECHNOLOGY (NIET),
SATTENAPALLI (M.D),
GUNTUR (Dist.), A.P.