Toward a Privacy Agent for Information Retrieval

Marc Juárez,* Vicenç Torra†
Institut d'Investigació en Intel·ligència Artificial, Consejo Superior de Investigaciones Científicas Campus, UAB s/n; 08193 Bellaterra, Catalonia, Spain

In this paper, we tackle the private information retrieval (PIR) problem associated with the use of Internet search engines. We address the desire for a user to retrieve information from the Web without the search provider learning about it. Traditional PIR protocols present two main shortcomings for their application: (i) They assume cooperation by the database, which is not affordable for a real-world search engine like Google and (ii) their computational complexity is linear in the size of the database, which is unfeasible in the case of the Web. More recent approaches relax PIR conditions to overcome these limitations and present some level of privacy. Mostly, they aim to distort server logs regardless of the loss of information that is involved. Server logs are used by search engines for profiling and, thereby, provide personalized results. This becomes a user's need given the growth of the Web and can also be used for targeted advertising. This study focuses on a noncooperative agent for private search that considers profiling as valuable data used for both sides of the search process. It is based on the assumption that the user's identity is formed by the union of various areas of interests or facets. Managing the HTTP connections properly, submitted queries are mapped to different server logs according to these facets. The rationale is that these logs cannot be used for tracing the user while they are still helpful for profiling. We present a personalized query classification approach based on the user's browsing history and to provide empirical results; we developed an attacking algorithm against the agent that shows that the disclosure risk is reduced. © 2013 Wiley Periodicals, Inc.

1. INTRODUCTION

Data privacy has been a research topic for a long time, with its roots in the field of statistics.[1] Over the past decade, along with the growth of the networked society and, more particularly, of online social networks, awareness of data privacy has risen within the data mining community. Data mining researchers turned their attention to the issue considering, first, privacy for standard databases, and later on for online social networks and Web services. Several scandals, such as the AOL search data leak in 2006, have helped to foster this research. Search engines collect

*Author to whom all correspondence should be addressed; e-mail: mjuarez@iiia.csic.es.
†e-mail: vtorra@iiia.csic.es.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 28, 606–622 (2013)

View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.21595

information from queries performed by the users in the form of server logs.[2] These logs include the queries of the user as well as data click-through. Then, these logs are used for profiling to provide more personalized results. Given a query, they rank the retrieved results according to user's personal preferences, such as main areas of interest, localization, age, or gender.[2–4] Nevertheless, it is obvious, as evidenced by the AOL scandal, that this information represents an important threat to user's privacy. The information we give to the search engine that improves the utility of the ranking makes us vulnerable and, with high probability, identifiable by the search provider.[5]

At present, a large number of tools[6–12] have been developed for privacy of search logs. Nevertheless, most of them are to be applied by the providers of the search engine. That is, they are developed in order that future releases of search logs by the providers do not cause new scandals. At the same time, a few approaches have also been developed to help the user to avoid the search engine from collecting such detailed information about him.

In this paper we focus on the latter approach. Previous research in this case includes GooPIR[13] and TrackMeNot,[14] as well as the work on user private information retrieval (UPIR)[15,16] as, for example, GooPIR and TrackMeNot are plug-ins to be installed in the browser to avoid the profiling by the search engine. GooPIR wraps the original query within an OR-Ring of bogus words that have similar frequency to the original ones. Then, it sieves results by cleaning those that contain bogus words in their snippets. TrackMeNot applies a similar strategy, but at a sessional level. It generates bogus queries, but instead of choosing words with similar frequency, it browses trending topics in RSS sources of newspapers.

All these approaches improve the privacy of the user by means of diminishing the quality of the profile. In the case of GooPIR and TrackMeNot, this is achieved by means of the noise introduced in server logs by the plug-in. In the case of Refs. 15, 16, this is achieved by means of mixing the user's server log with server logs of other users.

In this work, we follow a different approach. First, we consider profiling valuable, as it permits the user to access Web pages ranked according to his interests. Nevertheless, to ensure a level of privacy, we are of course interested in the search engine not being aware of all the queries a user makes. Second, our approach is to consider the user as a multifaceted individual and assume that the integration of all the facets of an individual is what makes him unique. Following these premises, our approach is to dissociate the different facets an individual has. To do so, we have designed the Dissociating Privacy Agent (DisPA). DisPA is implemented as a plug-in. It generates several virtual identities and contexts associated with them, one for each user's facet, and assigns each query to one of such contexts. As a result, the search engine can profile each virtual identity and personalize the queries according to these identities. For example, the search engine can track a series of queries of the virtual identity. At the same time, as the different virtual identities are not linked together, it is more difficult to link them to the real identity.

To illustrate, consider a user with the following two queries.

"buy new computer"
"blues music video"

The agent will assign each query to a different context. The first one to the context that has the identity for *Computers* and the second one to *Art*. To implement these virtual identities, identifiable information, for example, cookies related to the identities will be stored separately and used appropriately in conjunction with the HTTP protocol.

The structure of the paper is as follows: Section 2 reviews query classification. Section 3 describes the agent. Section 4 describes the empirical results. The paper finishes with some conclusions and some lines of future research.

2. QUERY CLASSIFICATION

One of the critical parts of our system is query classification (QC). This is used to separate the search logs in the server-side. Depending on the outcome of this classification, the query is mapped in a different server log. In Section 3.2, we show how we embed a simple QC model into a probabilistic model to achieve personalization. In this section, we review the QC problem.

The classification of queries turns to be more challenging than classification of other kinds of texts due to the following properties:

(i) They are short and sparse. For example, "ff," "ps."
(ii) They are noisy. For example, "facebook," "windows."
(iii) New words are emerging all the time. For example, "iphone5."
(iv) They lack context and are ambiguous. For example, "apple."

Additionally, their classification is subjective, depending on the interpretation and we do not have at our disposal large sets of labeled queries to train a classifier.

Because of these difficulties, given a query q and a subset of categories

$$C = \{c_1, \ldots, c_n\} \subseteq \mathcal{C},$$

where $\mathcal{C}$ is a taxonomy, the general problem of classification aims to determine the relevance of the i-th category for query q.

In 2005, the ACM KDD Cup, an annual competition hosted by ACM KDD conference, focused on Internet user search query categorization. About 40 teams from all over the world competed to classify a set of 800 queries that were labeled by three different people as classes of a taxonomy given by the organization. Since then, QC has received more and more attention in both academic and industry communities. Most existing QC research focuses on finding new features from queries to tackle the sparseness problem, or utilizing various Web resources to obtain additional training data.

The main difficulty of the ACM KDD Cup 2005 was the lack of training data. The winning solution[17,18] solved this problem by collecting documents from the Open Directory Project (ODP). The ODP, also known as *dmoz*, is the largest, most comprehensive human-edited directory of the Web and provides a taxonomy for Web pages. First, they created a mapping between the ODP taxonomy and the taxonomy provided by ACM to label the queries by comparing keywords from both structures. For example, if category c_1 in the ODP had some keywords in common with category c_2 in the target taxonomy, c_1 was mapped to c_2. Second,

they submitted queries to a search engine. By leveraging the search results, they extracted a vector of N features (topics) for each query that was used as an input for a Support Vector Machine (SVM) classifier. Finally, to train this classifier they produced training examples from ODP documents that were processed by stop word removal, stemming, and feature selection. Later works have followed the strategy of using the data of the ODP. In Ref. 19, the machine learning approach was changed to an information retrieval one. The authors used the ODP to build a search engine and classify queries by submitting them to it. Their results improved the solution developed for the KDD Cup 2005. We decided to adopt this last solution in our classification model due to its simplicity and because it makes classification applicable to a Web search scenario, where user's search has to be quickly bypassed.

Moreover, the techniques mentioned above that access other Web resources are not suitable to our case because they create new server logs that would not be protected. In contrast, techniques based on WordNet (e.g., Ref. 20) can be applied. WordNet is an ontology of the English language where nouns are organized in sets of synonyms, called *synsets*, and relations such as hypernymy and hyponymy are defined. The context of a query may be enriched adding synonyms and hypernyms of the terms. We implemented a WordNet enrichment using Apache's Lucene framework that was also used to build an inverse index of the ODP corpus.

3. DESCRIPTION OF THE AGENT

In this section, we describe the DisPA that we have implemented. As we introduced in the first section of this article, we consider the user as a multifaceted individual. We fix a collection of categories in the ODP taxonomy that represent the user's facets. For example, for a user who likes cinema and football, the category "Arts" in the ODP may represent the former interest and the category "Sports" the latter. DisPA assigns queries into categories through a probabilistic approach for query classification that will be explained in detail in Section 3.2. The main point of DisPA is to dissociate queries assigned into different categories mapping them to different logs at the server-side. To achieve that, the agent creates identities of fake users that we call *virtual identities*.

Virtual identities are formed by the set of session identifiers that are included in server logs:

1. Cookie ID
2. IP address
3. User-agent string

Then, the agent masquerades the user with a virtual identity by performing an HTTP spoofing; see Figure 1.

To keep a consistent session and prevent the server from detecting the agent, we define the connection context as data related to the HTTP connection that the search engine provider may use to detect a malicious behavior: All cookies from the server domain (even those that do not contain profile identifiers), the virtual identity, and other fields in the HTTP header like the *referral* field.

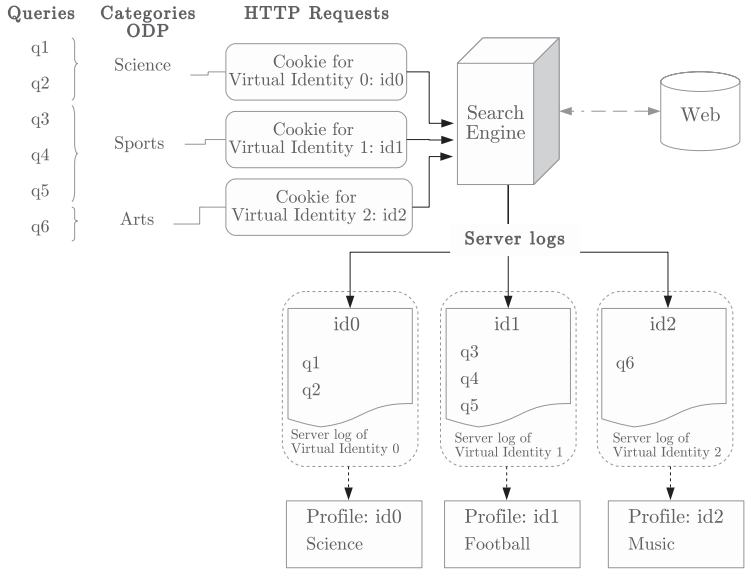

Figure 1. Representation of the spoofing and dissociation of queries.

Besides, there is more data that characterizes a search that we associate and store with a virtual identity. We define *context* as these data. When the user makes a query, DisPA classifies it into a certain category, then the agent bypasses the connection, and changes the context depending on the outcome of this classification. In DisPA, the context associated with each virtual identity includes the following:

Category: category of the ODP associated with this context.
Search context: search language, subdomain of the Web, date, and location of the search.
Connection context: data related to the HTTP connection. Note that virtual identities are included in the HTTP headers.
History of queries: queries that may fall into the category of this context.
Result pages of past queries: result pages returned from the search.
Browsing history: visited Web sites that may be classified into the category of this context.
Named entities: proper names of people and locations that appear in all queries of this context. This is further explained in Section 3.3.

In the following sections, we describe how the system manages the different contexts for a user, how queries are matched with a server log through personalized query classification, and how we deal with uncommon queries. Google search engine was used for developing and testing the agent.

3.1. Connection Bypass

Most Web sites need to remember the state of the connection to manage sessions and personalize contents. However, HTTP is a stateless protocol and extra

state-management mechanisms have to be added at the application layer.[21] The usage of cookies is the most common way to cope with this problem, but browser fingerprinting is also proved to be effective and is often used in terminals that do not support cookies.[22]

In particular, search engines implement these sort of techniques. From an analysis of Google's log retention policies[23] and the last published information by the company,[24] known session identifiers registered by Google in a server log are IP address, time and date, terms of the query, user–agent string, and cookie ID. The same policy report reveals that cookies are the main mechanism used by Google to track sessions.

As the Google's director product management, Jack Menzel clarifies in this interview,[25] Google invests a lot of effort in personalizing searches of users that do have a Google account, but resources dedicated to users that are not logged in are much lesser. The data of these users are only stored for 180 days, and finding patterns becomes much more difficult. The agent focuses on the latter case because we assume that a logged user allows his identification.

Nevertheless, the personalization of advertisements is almost the same for all users. The *id* cookie from doubleclick.net is associated with a profile in the *display network*, the community of sites where Ad-words advertisements may appear. To offer more personalized advertising, Doubleclick cookies are used to track cross-domain visits and profile users.

To generate a virtual identity for each context, we need to generate different cookies, as well as hide, or mask, other identifying elements. We discuss these issues below.

There are two kinds of ID cookies that we take into account. First, we have the cookies from Google's domain. To generate them, the agent establishes a new HTTP connection with an empty *cookie* field in its header. This way the server interprets that a new user is connecting for the first time and issues two cookies: *PREFS* and *NID*,[26] which contain the unique identification number of the user's profile. These cookies are stored in the virtual identity of a context associated with a given category.

Second, we have cookies from Doubleclick.net. The acquisition of the *id* cookie is more difficult because the contents of the page www.google.com/ads/preferences are loaded dynamically and the link to enable personalized advertising contains a hash that changes for each new connection. We have to read the Document Object Model (DOM) structure of the page when it is fully loaded and make the request to the Doubleclick Web server.

Regarding the IP address, there is much more ambiguity while tracking a user, because there may be hundreds or thousands of people that are surfing behind the same router. However, it gives information about the location of the user. Upper layer protocols like ToR[27] may be considered as a complement of the agent to mask the IP address.

The user-agent gives other information,[22] which is read by Web servers to offer friendly content adapted to the characteristic features of the device. The approach followed here is to change the browser's agent to a more generic one, for

example[a]:

> Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11
> (KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11

However, this string gets obsolete very fast; there are upgrades of the software almost yearly.

The agent also ensures that a query is never sent twice. The rationale is that very repetitive queries could lead to identification. An example is *navigational queries,* which are queries that the user performs to reach another Web site rather than to look up information. It turns out to be more comfortable to search the name of the site than to memorize the Web address. For instance, it is very common to search "facebook" to get to www.facebook.com. This also happens with job's sites, college's sites, etc. Therefore, it is reasonable to set some kinds of relation between the user and the site. The solution to this problem is to implement a caching system in the agent so that every time a query is performed and its results are retrieved, they are stored into disk. Then, when the same query is detected the corresponding results are fetched and shown.

As summarized in Figure 2, the basic algorithm of DisPA is as follows: Check whether the query has already been sent, in that case we load the corresponding stored results page and the agent has finished. Otherwise, the agent classifies the query into the taxonomy and checks whether the resulting category has already an associated context. If the context is not found, a new virtual identity for a new context is generated and stored as explained above. Otherwise, this last step is omitted and the agent loads the connection context of the context associated with that category. Then, the agent submits the query. Finally, new issued cookies and changes in the connection context are written to disk. The results page retrieved from Google is stored and shown to the user.

Besides, in the background, DisPA extracts metatags of all visited pages by the user to build a query that represents their contents and submits it to our classifier. The outcome of the classification is stored as a weight in the resulting category. These weights give us an approximation of the distribution of interests of the user, which is used for Personalized Query Classification (PQC). This is further explained in Section 3.2.

As an example, imagine a scenario where a user wants to submit the query q_1 ="basketball match" using the agent for the first time. When the browser is executed, the user clicks on the plug-in icon placed in the add-on Firefox bar to get the DisPA's search page. This page has listeners waiting for the event of sending the main form. When the query is submitted, the agent has to create a new context. Suppose that q_1 is assigned to "Sports," the agent generates cookies as explained above, stores them in the "Sports" directory and loads them on the browser. Then the user-agent is changed and the query is submitted. The Web pages returned with

[a]This user-agent string turned to be the most generic from a survey that we did and accessible at http://www.iiia.csic.es/∼mjuarez/results.html. Results showed that this user-agent string was used 11 times out of 87.

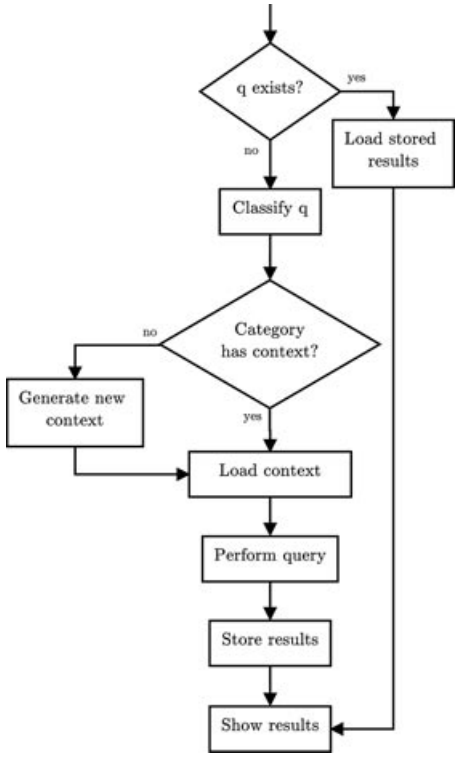

Figure 2. Basic bypass flowchart.

the results are also written to disk in a cached directory to deal with repetitive queries.

Now imagine that the user searches for q_2 ="Michael Jordan." The query will be classified into "Sports" again, but this time the agent will find an existing context. Finally, if q_1 is submitted again these steps are also skipped and results that were stored in the first search are loaded.

3.2. Personalized Query Classification

To take advantage of personalized search, the agent performs a personalized classification locally and attempts to use virtual identities and contexts that describe the facets of the user as well as possible.

Given a query q_0, a simple query classification, that does not take into account the user's preferences, may be achieved by:

$$c = \mathit{classify}(q_0) = \arg\max_{c_i \in C} \; p(c_i \mid q_0).$$

In our approach, the determination of $p(c_i \mid q_0)$ uses ODP. We search ODP in a local ODP-based search engine. If $h_i(q)$ is the number of hits in the i-th category when

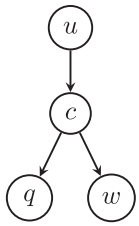

Figure 3. Graphical model representation for PQC.

we search query q_0, then we estimate $p(c_i \mid q_0)$ by

$$p\left(c_i \mid q_0\right) = \frac{h_i(q_0)}{\sum_i h_i(q_0)}. \tag{1}$$

The probabilistic model used to achieve PQC presented in this article is similar to the one presented in Ref. 28. We contribute with a novel approach based on inferring the user's interests by means of the history of navigation. The idea is that DisPA, as a plug-in in the user's browser, has direct access to much more data and may take advantage of it to infer the user's traits.

Assume that a user u wants to search for the information related to query q. The search engine may interpret q as being from a set of categories. Independently of this, the user accesses Webs w that could be classified in this set of categories. Thus, w depends on c. In Figure 3, we see the graphical representation of this model.

Formally, QC aims to estimate $p(c \mid q)$, namely the conditional probability of searching information about the category c given the query q. In contrast, PQC also takes into account the user u, so the expression $p(c \mid q, u)$ is considered. $p(c \mid q, u)$ is the probability that the query q belongs to category c for the user u. Using the joint probability,

$$p(c \mid q, u) = \frac{p(c, q, u)}{p(q, u)}. \tag{2}$$

Let us now develop this expression. The numerator can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$p(c, q, u) = p(c)\ p(q \mid c)\ p(u \mid q, c). \tag{3}$$

The first factor is the prior probability of the category c that we approximate by the number of documents per category in the ODP. The second factor is the probability of generating the query q from c. The third factor is the probability of occurrence of the user u conditioned on the category c and query q. As it is discussed in Ref. 28, this last factor can be simplified by assuming that the user has stable interests that

are independent of the current query q,that is,

$$p(u \mid q, c) = p(u \mid c).$$

This assumption holds for long-term and short-term searches. For instance, from a long-term perspective, a researcher may have stable interests in information related with his field. As for the short-term perspective, any user has a fixed set of topics related to the information that he is looking for within the search session.

Taking these expressions into account, Equation 2 results in

$$p(c \mid q, u) = \frac{p(c)\ p(q \mid c)\ p(u \mid c)}{p(q, u)}. \tag{4}$$

Using the Bayes rule, we obtain $p(q \mid c)$ from $p(c \mid q)$ using Equation 1. That is,

$$p(q \mid c) = \frac{p(c \mid q)\ p(q)}{p(c)}. \tag{5}$$

Finally, we need to estimate $p(u \mid c)$. As discussed in Ref. 29, the majority of users are reluctant to give explicit feedback information about their search preferences. As described in Ref. 28, personalization without user registration is normally achieved by search providers from analysis of

(1) history of queries
(2) data click-through

The idea from point (1) is that historical queries submitted by a user can be used to help learning preferences:

$$p(u \mid c) \approx p(q_1, \ldots, q_s \mid c),$$

where q_i for $i = 1, \ldots, s$ are queries performed within a session. Nevertheless, sessions containing the current query may not reflect all search interests of a given user.

In point (2), from the user's point of view, queries belong to one of the candidate categories and clicked pages must fall in the same category. This is the approach followed in Ref. 28. They built a collaborative model for learning user's preferences, assuming that users that clicked the same links for the same query tend to have similar interpretations for that query. However, this assumption does not always hold true. It is common that user searches over the results list by click-and-try and clicked links may not contain the searched information.

As we said, in contrast to search providers, the agent is executed locally and can exploit more information. Our approach is to consider that the history of visited pages provides a more detailed description of the user's preferences than the history of queries. For instance, visited pages include those pages that the user has accessed

from typing their addresses. This is expressed as

$$p(u \mid c) \approx p(w_1, \ldots, w_t \mid c). \tag{6}$$

That is, the probability of occurrence of user u, given the category c, is approximately the probability of the history of visited pages $w_1, \ldots, w_t$ in given category c. The above equation shows that we can treat this problem as a classification problem. Here, the difference from previous work is that we classify all Web pages that the user visits, not only those that are clicked from the search results. The idea is that there are a lot of Web sites that are accessed by a direct address or by a chain of clicked links that are not collected in the data click-through. As introduced before, we already have a taxonomy for Web site classification, the ODP. We classify Web sites reading their metatags: keywords, description, title, or by submitting their content as a query.

Let us now reconsider Equation 4 in light of previous equations. Now we can approximate this probability as

$$p(c \mid q, u) \approx \frac{p(q)}{p(q, u)}\, p(c \mid q) p(w_1, \ldots, w_t \mid c). \tag{7}$$

Note that we can ignore those factors that do not depend on the class c because they will be constant for all classes. So, the classifier is

$$\mathit{classify}(q_0) = \arg\max_{c_i \in C}\ p(c_i \mid q_0)\, p(w_1, \ldots, w_t \mid c_i). \tag{8}$$

As we said, we estimate this classification using the ODP. On the one hand, we classify queries by using the local engine, and, on the other hand, we classify visited pages on real time. Note that the ODP is very suitable for the latter since it is a taxonomy for Web sites. Taking all this into account, the classifier can be expressed in terms of the search as

$$\mathit{classify}(q_0) = \arg\max_{c_i \in C}\ h_i(q_0)\, \frac{v_i}{|c_i| + v_i}, \tag{9}$$

where v_i is the number of visited sites classified in the ith category of the ODP.

3.3. Filtering Uncommon Queries

The AOL scandal evidenced that the history of queries, by its own, could lead to the identification of a user.[5] Our approach is to consider the less frequent queries among all users of the search engine as those that give more information to possible attackers. We assume that people fond of rare topics are easier to identify. We take rare queries as those that have fewer results. We approximate the number of results of a Google's query with the number of results that the ODP search returns. We take this number of results as the absolute frequency of the query. We assume that

queries that have more results are more frequent than others. Following the notation in the preceding section, this corresponds to

$$f(q) \approx \sum_i h_i(q).$$

We set a threshold and, in case that $f(q)$ is under it, the agent will generate a new virtual identity only for this query.

As evidenced by the AOL scandal, queries that contain named entities are an example of uncommon queries that lead to identification, in particular, personal names and names of locations. To detect them, we use the Stanford's CoreNLP library for Named Entity Recognition (NER). The agent generates a new virtual identity for each new combination of category and collection of named entities. The rationale is that DisPA still dissociates the user's facets considering only queries containing the same collection of named entities. As we explained in Section 3, contexts have associated these sets with named entities.

On the one hand, this approach argues that server logs contain less infrequent terms in common. As we will see in the following section, it is more difficult to rebuild the original server log from dissociated logs by DisPA and, hence, there is less disclosure risk. On the other hand, this dissociation decreases personalization since server logs contain less data to build profiles. Besides, NER in queries has the same problems caused by the special characteristics of queries reviewed in Section 2. Other techniques like generalization of named entities are proposed in Section 5.

4. EMPIRICAL RESULTS

We have implemented the DisPA agent as explained in the previous sections of this paper. The lack of public sets of queries makes difficult the evaluation of the degree of personalization achieved by the agent as well as the comparison between using and not using the agent. A superficial check with a set of 803 queries from one user of the AOL data set seems to give appropriate Web pages. Another experiment with a set of 2743 queries of another AOL user showed that the difference between personalized and nonpersonalized search only makes a difference on the order of two results (from first to second place in the ordered list of results). Nevertheless, a complete analysis of personalization is out of the scope of this paper. Because of that, in this section we focus on disclosure risk.

We developed an attacking algorithm against the agent. Such an algorithm could be used by a presumed attacker that has access to the collection of logs in the search engine's server. It aims to rebuild the original user's server log from partial logs generated by DisPA. We assume that there are terms that are more common in the user's searches than in the searches of other users. We also suppose that a subset of these terms does not fit into a concrete category of the taxonomy and may be spread over all the partial logs. As mentioned in Section 3.3, an example of these terms is named entities.

The idea is to express logs as vectors using the term frequency–inverse document frequency (tf-idf) scheme. The rationale is that tf-idf reflects the importance of a term in a log, offset by its frequency in the collection of all logs. Then, the algorithm clusters the vectorial space using the DBSCAN clustering algorithm with the cosine similarity as a distance. The algorithm starts with one or more seeds corresponding to logs that are known to be of the user. At the end, all clusters that contain a seed are joined into one unique cluster that represents the original log (i.e., the union of all logs generated by the agent). We propose to measure the user's data disclosure risk as the $F1$ score of the binary classification as defined by the property of being part of the original log. Note that since server logs have a different number of queries, a F1 score of a small sample of logs can have a bias. For this reason, we used queries in examples instead of directly using the examples of the clustering: True positives are queries of the original log that fall in the final cluster, false positives are queries of other users that fall in the final cluster, true negatives are queries of other users that fall out of the final cluster, and false negatives are queries of the original log that fall out of the final cluster.

4.1. Experiments

For these experiments, we used the AOL-released data set of queries treating the queries of the users as server logs.

The empirical part is divided in three experiments. For the first experiment, we picked a sample of 20 users and, after applying the DisPA algorithm without filtering uncommon queries, we applied the attacking algorithm taking a random dissociated log as a seed. For the second experiment, we added to these logs the one of Thelma Arnold, the first AOL user being identified. This time we chose Thelma Arnold's log as the target log because it contains named entities like "Arnold" or "Lilburn" in queries that would fall in different categories. We repeated the first experiment with this log to find if the attacking algorithm performs better. To initialize the clustering algorithm, we took as seed the dissociated log corresponding to the class "Arts," one of the largest dissociated logs. The justification is that it is more likely that the attacker finds the information that he knows about the user in the largest log. Also, the attacking algorithm performs better with seeds that contain more rare terms that are common for all dissociated logs, and larger logs are more likely to contain these terms. For the third experiment, we repeated the second experiment but using the filter of uncommon queries and treating queries with named entities as described in Section 3.3.

4.2. Analysis

For the first experiment, we measured the recall, the precision, and the F1 score of the clustering. In Figure 4, we show these measures for different values of ε. This number is a parameter of the clustering algorithm that defines the neighborhood of a cluster. We see that for small values of ε, the algorithm has maximum precision because the final cluster only contains the seed. Then, all the queries of the seed are queries of the target log that fall in the final cluster (true positives) and since there

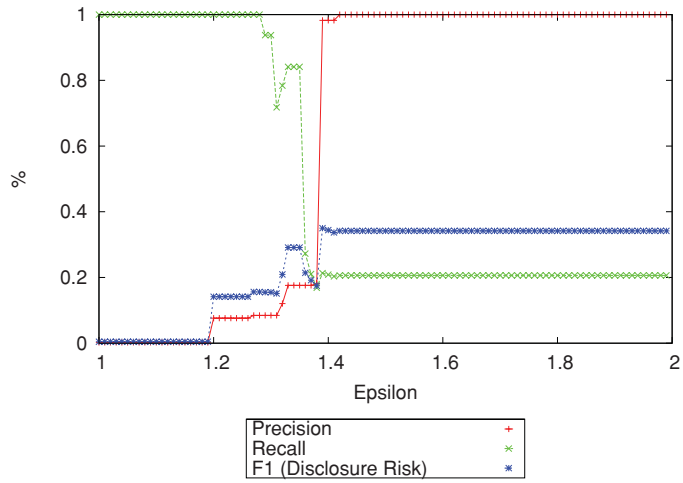

Figure 4. Evaluation of the clustering using DisPA.

is no other server log in the final cluster, there are not queries of other users in the final cluster (false positives). In contrast, the recall is very low because there are a lot of server logs of the user that fall out of the final cluster (false negatives). This is translated to a low F1 score and hence a low disclosure risk. As we increase ε, there are more profiles that fall into the final cluster. Nevertheless, this server log is well dissociated by DisPA and the algorithm, for the given seed, jumps directly to the situation where the whole collection of server logs falls into the final cluster. This means that user's logs cannot be linked using the algorithm with this seed because they do not have enough rare terms in common.

In Figure 5, we show the results of the second experiment. This time we used Thelma Arnold's log as the target server log. As we see in this figure, for $\varepsilon = 1.39$, we have a good recall and a good precision, which means that almost all dissociated logs of Thelma Arnold fall into the final cluster and logs of other users fall out. The algorithm has linked most of the dissociated logs, and, thus, we have a high disclosure risk. If we compare Thelma Arnold's log with the one from the previous experiment, we see that she searches for terms such as "Arnold" or "Lilburn," which are uncommon terms and fall in different categories when submitted to our local classifier. Note that one flaw of this algorithm is that the optimal epsilon must be known a priori by the attacker, other clustering algorithms like OPTICS that do not need this parameter could be considered.

Finally, for the third experiment we took the same parameters for the attacking algorithm, but this time we used the filter for uncommon queries described in Section 3.3. In Figure 6, we see how disclosure risk is almost constant for the same step of ε taken before. Note that for $\varepsilon = 1.9$, disclosure risk increases. On the one hand, we see that precision is maximum for greater values of ϵ, which means that the cluster contains all dissociated server logs. On the other hand, recall has decreased, which means that it also contains a lot of noise. For this reason, we know that the

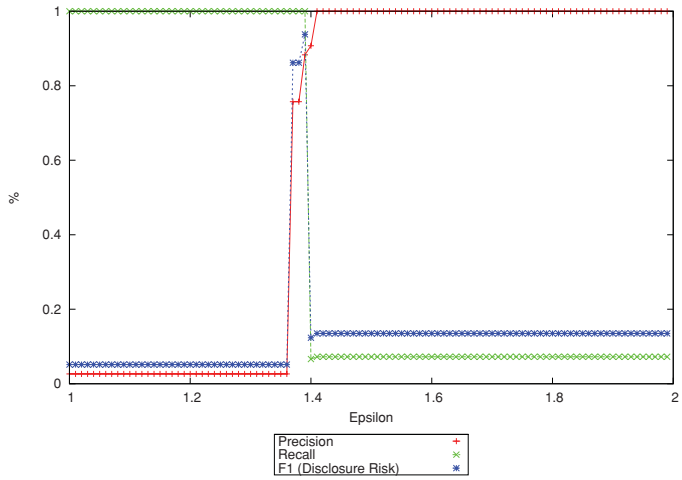

Figure 5. Evaluation using Thelma Arnold's log as target.

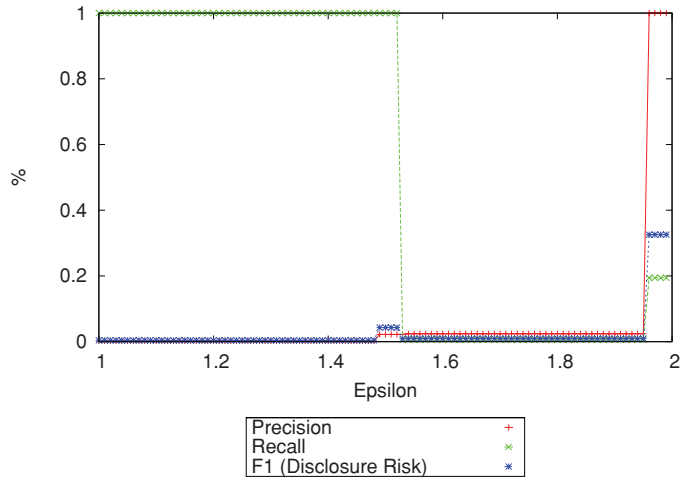

Figure 6. Evaluation using the filter for uncommon queries.

algorithm will not perform better if we increase the neighborhood distance. This means that logs cannot be linked because uncommon terms have been separated successfully in different logs.

5. CONCLUSIONS AND FUTURE WORK

The main contribution of this study is the implementation of an agent that provides less disclosure risk in search engines with an admissible time of response

of search. Empirical results have proved that disclosure risk is much lower with this method. In addition, we present a new model for PQC, based on the analysis of the user's browsing history that presumably preserves personalization of search engines. However, DisPA has some drawbacks that future research may deal with. Future work following this study can focus on improving the personalized classification. We think that building the user's taxonomy dynamically, instead of fixing a collection of classes of the taxonomy, may solve problems of undefined interests that fosters attacks such as the one described in Section 4. In personalization, we should also consider vertical searching. Vertical searching refers to the process of refining queries by adding more terms. This personalization may take into account short-term preferences within a session. One of the issues that arises is how to detect sessions as a collection of queries that look for the same information.

Following decreasing disclosure risk, one could consider the generalization of named entities using an ontology like WordNet. For instance, if someone searches for "lilburn dentists" the agent could generalize "Lilburn" to "Atlanta" or "Georgia." Information loss would be greater, but then it would be possible to measure it by the differences between search results pages.

Besides, the attacking algorithm described in Section 4 can be tested with different clustering approaches like sequential clustering described in Ref. 30. The similarity measure could be improved by applying a boost in the tf-idf scheme using a dictionary of terms that differentiate the user from the others. In addition, other measures of disclosure risk may be defined comparing clusters between clustering of DisPA and nonfiltering DisPA. For instance, if dissociated logs in the former fall into several clusters in the latter, disclosure risk is lower than if all fall in the same cluster. The Jaccard index could be used to measure differences between clusters of both classifications.

Acknowledgments

This study received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 262608.

Partial support by the Spanish MEC projects ARES (CONSOLIDER INGENIO 2010 CSD2007-00004), eAEGIS (TSI2007-65406-C03-02), and COPRIVACY (TIN2011-27076-C03-03) is also acknowledged.

References

1. Willenborg L, Waal TD. Elements of statistical disclosure control, vol. 155. New York: Springer; 2001.
2. Grimes C, Tang D, Russell DDM. Query logs alone are not enough. In: Workshop on query log analysis at WWW; April 2008.
3. Zamir OE. Personalization of placed content ordering in search results, 2005. Patent no. 20050240580.
4. Kenneth WT, Lee DL. Deriving concept-based user profiles from search engine logs. IEEE Trans Knowl Data Eng 2009;99(1):969–982.
5. Barbaro M, Zeller T. A Face is exposed for AOL searcher no. 4417749. 2006.
6. Adar E. User 4xxxxx9: anonymizing query logs. In: Proc Query Log Analysis Workshop, Int Conf on World Wide Web; 2007.

7. Cooper A. A survey of query log privacy-enhancing techniques from a policy perspective. ACM Trans Web 2008;2(4):19.
8. He Y, Naughton JF. Anonymization of set-valued data via top-down, local generalization. Proc VLDB Endowment 2009;2(1):934–945.
9. Hong Y, He X, Vaidya J, Adam N, Atluri V. Effective anonymization of query logs. In: Proc 18th ACM Conf on Information and Knowledge Management, CIKM '09 New York, NY: ACM; 2009. pp 1465–1468.
10. Navarro-Arribas G, Torra V. Tree-based microaggregation for the anonymization of search Logs. In: 2009 IEEE/WIC/ACM Int Joint Conf on Web Intelligence and Intelligent Agent Technology, Milan, Italy; 2009. Vol 3, pp 155–158.
11. Miller G. WordNet-about us. WordNet. Princeton University (2009); 2010.
12. Erola A, Castellà-Roca J, Navarro-Arribas G, Torra V. Semantic microaggregation for the anonymization of query logs using the open directory project. SORT—Stat Oper Res Trans 2011;35(1):41–58.
13. Domingo-Ferrer J, Solanas A, Castellà-Roca J. $h(k)$-private information retrieval from Privacy-Uncooperative Queryable Databases; 2008.
14. Howe DC, Nissenbaum H. TrackMeNot: resisting surveillance in web search; 2008.
15. Domingo-Ferrer J, Bras-Amorós M, Wu Q, Manjón J. User-private information retrieval based on a peer-to-peer community. Data Knowl Eng 2009;68(11):1237–1252.
16. Stokes K, Bras-Amorós M. Optimal configurations for peer-to-peer user-private information retrieval. Comput Math Appl 2010;59(4):1568–1577.
17. Shen D, Pan R, Sun J, Pan J, Wu K, Yin J, Yang Q. {Q2C@UST}: Our winning solution to query classification in {KDDCUP}2005. SIGKDD Explorations 2005;7:100–110.
18. Shen D, Pan R, Sun J-T, Pan JJ, Wu K, Yin J, Yang Q. Query enrichment for web-query classification. ACM Trans Inform Syst 2006;24(3):320–352.
19. Ullegaddi P, Varma V. Simple unsupervised query categorizer for web search engines; in Proceedings of ICON-2010: 8th International Conference on Natural language Processing. Macmillan Publishers. 2011.
20. Montoyo A, Palomar M, Rigau G. Wordnet enrichment with classification systems. In Proc. of the NAACL Workshop of WordNet and Other Lexical Resources: Application, Extensions and Customisations; 2001, 101–106.
21. Barth A. HTTP state management mechanism. RFC 6265 (Proposed Standard); Apr 2011.
22. Eckersley P. How unique is your web browser? Technical report, Electronic Frontier Foundation; 2009.
23. Toubiana V, Nissenbaum H. Analysis of Google logs retention policies. J Privacy Confident 2011;3(1).
24. Google. Key Terms - Policies and Principles; Apr. 2012. http://www.google.com/intl/en/policies/privacy/key-terms/\#toc-terms-server-logs.
25. Menzel J. How Google does personalization with Jack Menzel; Jan. 2011. http://www.stonetemple.com/how-google-does-personalization-with-jack-menzel/
26. Milly. Anonymizing Google's cookie; 2004. http://www.imilly.com/google-cookie.htm
27. The Tor Project. Facebook, a fun resource or invasion of privacy.; Sept. 2002. https://www.torproject.org
28. Cao B, Sun J, Xiang E, Hu D. PQC: personalized query classification. In: Proceedings of the 18th ACM conference on information and knowledge management; 2009. pp 1217–1225.
29. Carroll JM, Rosson MB. The paradox of the active user. In: Interfacing thought: Cognitive aspects of human-computer interaction, Cambridge, MA: Bradford Books/MIT Press; 1987. Chap 5, pp 80–111.
30. Miyamoto S, Arai K. Different sequential clustering algorithms and sequential regression models. In: Proc IEEE Int Conf on Fuzzy Systems, Jeju Island, Korea; 2009. pp. 1107–1112.