Working at the Web Search Engine Side to Generate Privacy-Preserving User Profiles

David Pàmies-Estrems[a], Jordi Castellà-Roca[b], Alexandre Viejo[b]

[a]*Aneris, Raval Sta. Anna 43, 2n 2a, E-43201 Reus, Spain*
E-mail: david.pamies@aneris.cat
[b]*Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, UNESCO Chair in Data Privacy, Av. Països Catalans 26, E-43007 Tarragona, Spain*
E-mail: {jordi.castella, alexandre.viejo}@urv.cat

Abstract

The popularity of Web Search Engines (WSEs) enables them to generate a lot of data in form of query logs. These files contain all search queries submitted by users. Economical benefits could be earned by means of selling or releasing those logs to third parties. Nevertheless, this data potentially expose sensitive user information. Removing direct identifiers is not sufficient to preserve the privacy of the users. Some existing privacy-preserving approaches use log batch processing but, as logs are generated and consumed in a real-time environment, a continuous anonymization process would be more convenient. In this way, in this paper we propose: i) a new method to anonymize query logs, based on k-anonymity; and ii) some de-anonymization tools to determine possible privacy problems, in case that an attacker gains access to the anonymized query logs. This approach preserves the original user interests, but spreads possible semi-identifier information over many users, preventing linkage attacks. To assess its performance, all the proposed algorithms are implemented and an extensive set of experiments are conducted using real data.

Keywords: Web search engines, Privacy, Query logs, Data stream, Data monetization.

1. Introduction

Nowadays, surfing the Web usually requires using a Web Search Engine (WSE) as main entry point. Individuals interact with WSEs by submitting

search queries that contain certain keywords related to the topics they are looking for. WSEs, then, retrieve and present, with a minimal response time, an accurate list of web sites that tackle those topics. The popularity of WSEs has grown with the number of websites present on the Internet. According to a recent study (Netcraft, 2016), the number of websites is nearly doubling every three year period; therefore, it can be assumed that WSEs such as Google or Bing will continue to be essential players in the next years.

When a WSE looks for the requested information among its indexed web pages, it also stores the submitted query (i.e., the keywords) and some metadata (e.g., date of the query, some identifiers of the query issuer, which specific search result was selected by the issuer, etc.). The files that contain all the recorded queries and their related metadata are generally referred as *query logs* (Chau et al., 2005). As a result, everyone who uses a WSE is disclosing personal data, such as personal characteristics and preferences, and enabling WSEs to compile those query logs, which are part of the huge bunch of data that has been recently designated as *Big Data*.

Turning this data into knowledge is a main interest for many companies. Knowledge is gathered using several techniques related to data mining and machine learning. The result of these processes is the extraction of valuable information that enable those companies to obtain economic benefits. This process is known as *data monetization* (Saagar, 2014) and it was recently put on the spotlight by the U.S. Federal Trade Commission when this organization published a report about data collection and use practices of the most relevant Data Brokers (U.S. Federal Trade Commission, 2014).

Once the query logs are generated, they are processed and analyzed in order to build user profiles. In the literature, a user profile is generally considered a set of well-defined categories of interests (e.g., science, health, society, sports, etc.) with a certain weight assigned to each one according to the evidences generated by the corresponding user and how they have been classified under each category (Viejo et al., 2012). When focusing on WSEs, search queries are the evidences, and the amount of queries from each user classified under each category reflects the related weight.

WSEs process and analyze query logs and related user profiles in order to perform the following services:

- *Personalization*: Providing relevant results to their users' needs, where relevant links are placed in the first positions of the returned results. This is achieved by analyzing the past queries submitted by users; this

knowledge allows WSEs to contextualize and disambiguate next queries (Shen et al., 2007). In this way, if the user searches for "Mercury" and her profile indicates that she is interested in "Astronomy", the WSE will put the results that correspond to the planet Mercury in the first pages (instead of the chemical element).

- *Improved web search*: By knowing the frequencies of most formulated queries and most selected results, WSEs are able to improve the ranking algorithms (Agichtein et al., 2006) and to suggest reformulated queries that can add specificity to the user's initial query (Jones et al., 2006). Suggestions can be offered while the user is typing her query or also after retrieving poor search results for a query submission (Cooper, 2008). Following with the example of the term "Mercury", if a user inputs just this term in the text-field of Google's WSE, it will suggest as better alternative queries: "Mercury - Planet", "Mercury - Element", "mercury poisoning" and "mercury marine" among others. These alternative keywords are expected to retrieve more accurate results to the user.

- *Marketing*: Characterizing general user profiles, user behavior and user search habits, it's possible to improve keyword advertising campaigns and extract market tendencies among others (Brenes & Gayo-Avello, 2009; Poblete et al., 2007; Korolova et al., 2009). More concretely, search analytics is one of the cornerstones of the so-called *Search Engine Marketing* and it is in charge of using search data to investigate particular interactions among Web searchers, the search engine, or the content during searching episodes (Jansen, 2006). For example, Google Trends[1] is a service that exploits this kind of data. In particular, Google Trends shows how often a particular term is searched according to the total search-volume across various regions of the world.

- *Research*: As stated in (Richardson, 2008), query logs contain information that would never be available to researchers using conventional data-collection techniques. For example, a medical researcher might discover that people with asthma tend to wear wool, or live in areas with coal power plants; a sociologist may study how ideas spread

[1]Google Trends. `https://www.google.com/trends/`

from one person to an entire community, or may investigate the differences between the interests that individuals claim to have during face-to-face interviews and the real interests that their search queries reveal (Tancer, 2008); a political scientist might learn about democracy by studying the evolution of political searches by users in a developing country; and a computer scientist may study and analyze new Information Retrieval (IR) algorithms via a common benchmark query log (Bar-Ilan, 2007). Last but not least, query logs also enable researchers to ask questions that would normally require going backward in time. For example, a medical researcher might study people diagnosed with diabetes today to find out what their primary symptoms were six months ago. Asking them directly, once they learn they have diabetes, may result in subjective bias (Richardson, 2008).

In addition to those benefits, building and exploiting query logs may lead to serious privacy problems as well. More specifically, the keywords of each query and the related metadata may provide to anyone who has access to the logs with sensitive information from the users such as behaviors, habits, interests, religious views, sexual orientation,etc. Even worse than that, some query contents may contain identifiers and quasi-identifiers which may allow to link a certain query with a real person. An example of this situation happens with the so-called vanity searches (Soghoian, 2007) in which an individual looks for its own name on the Internet.

Although query logs could be protected prior to their publication, there is no absolute guarantee of anonymity, as the combination of modified data may disclose enough information to re-identify some users (Poblete et al., 2007; Jones et al., 2007). As an example, there is one well-known case, the AOL scandal, in which around 36 million queries performed by AOL's costumers were publicly disclosed. Although records were previously de-identified, it was possible to identify some users from the disclosed query logs and other sensitive information was exposed (Barbaro & Zeller, 2006). This case ended up with an important damage to AOL users privacy and to AOL itself, with several lawsuits against the company (Mills, 2006).

Therefore, in order to get viable data monetization, better tools capable of modifying query logs by limiting the privacy disclosure risk but preserving as much data utility as possible should be provided. More specifically, in this paper, we propose using a server side software capable of processing queries in real time and building anonymized query logs that still retain enough

data utility to allow its monetization. As a result, WSEs may then offer the protected query logs to external organizations for data monetization purposes while keeping the real query logs in a safe place; otherwise, WSEs may also decide to only keep protected logs, getting in turn a lower risk of information disclosure in case of a direct attack.

The rest of this paper is organized as follows: in Section 2, we summarize the most relevant research published up to date. In Section 3, we define the method used to implement the system. In Section 4, we analyze the results. In Section 5, we conclude and point out some future lines of research.

2. Previous work

To avoid record-linkage attacks, the concept of k-anonymity is proposed in (Samarati & Sweeney, 1998). A k-anonymized dataset has the property that each record is indistinguishable from at least $k-1$ other records, no individual can be re-identified with probability exceeding $1/k$ through linking attacks alone. Currently there are two main approaches to protect users' records: batch or real-time processing.

2.1. Batch processing

Historically, these were the first proposed methods. Because all the logs are stored, it's possible to remove the queries that disclose the identities of the individuals by means of basic statistic or semantic methods. We next detail the different methods that belong to this category considering two main subcategories: i) query removal; and ii) statistical disclosure control (SDC) and semantics.

2.1.1. Query removal

As shown in (Cooper, 2008), several approaches dealing with the query log anonymization problem based on query removal or hashing can be found in the literature. More concretely, some protocols and methodologies simply remove old query sets assuming that query logs will not be large enough to enable identity disclosure, however, this assumption does not take into account the existence of highly identifying queries (Barbaro & Zeller, 2006).

A more appropriate approach suggests deleting only infrequent queries (Adar, 2007), assuming that those are more likely to refer to identifying or quasi-identifying information. However, this is difficult to achieve due to the fact that the vast majority of queries occur a small number of times and, in

any case, it can be quite challenging to select the proper deletion threshold. As a result, this method may result in eliminating a substantial amount of non-identifying but useful queries (Beitzel et al., 2004).

Other removal-based methods focus on removing identifying data associated to queries and private information found within the given keywords (e.g. SS numbers, credit cards, addresses, etc.) (Center for Democracy and Technology, 2007). However, as explained in the Introduction, the AOL incident demonstrates that combining the remaining no-identifying data may be enough to disclose the identities of the individuals.

More elaborated approaches in this field focuses on removing those queries that end up with the user clicking common URLs, considering that those queries may be dependent, i.e. quasi-identifiers (Poblete et al., 2007; Korolova et al., 2009). Generally, these methods represent query logs as a graph in which queries are nodes and are connected by an edge if the intersection of their clicked URLs sets is not empty. The anonymization process consist on removing all queries that: i) return less than k documents; ii) contain part of target URL; and iii) contribute to a non-zero density of the query graph.

2.1.2. Statistical disclosure control (SDC) & Semantics

More recent approaches rely on SDC methods to anonymize query logs while retaining the maximum level of information by means of minimizing the amount of query removal (Navarro-Arribas & Torra, 2009; Hong et al., 2009). They group users with similar query logs together and, then, their queries are replaced by a representative query. According to that, stored queries are transformed in order to minimize the disclosure risk.

Due to the dimensionality and unbounded nature of query logs, some authors such as (He & Naughton, 2009) have proposed to anonymize the set of queries made by an user by means of generalizing her queries using a knowledge base such as WordNet (Miller, 1995). The problem of this approach is that a query can be meaningless in a generic dictionary, but it might be dangerous according to a more specific one. Pioneer work on this field such (Terrovitis et al., 2008) generalizes groups of input queries to a common conceptual abstraction (e.g. *sailing* and *swimming* to *water sports*), until users who performed those queries become k-anonymous.

2.2. Real-time processing

The approaches that have been introduced previously work only with static data. This implies that, if applied to anonymize query logs, these

sources of data must be "closed" before applying any procedure. According to this, and depending on their size, two main options are considered: i) the query logs to be anonymized contain all the queries received from time 0 until now; or ii) the query logs to be anonymized correspond just to a short and limited period of time.

In the first option, the anonymization method in use will deal with a huge quantity of data; even worse, query logs are expected to grow daily and, hence, each anonymization process will be required to process bigger amounts of data. In order to show the relevance of this problem, it has to be noted that usual anonymization methods that provide k-anonymity in databases have shown significant limitations when dealing with large quantities of data. For example, well-known SDC techniques based on generalization or microaggregation suffer from a very high cost when performing the partition of the database, in this way, authors have reported quadratic costs in this steps that clearly disqualifies these techniques for the volume of information that is considered in this case (Soria-Comas & Domingo-Ferrer, 2015). Being more specific, the authors in (Batet et al., 2013) apply semantic microaggregation to anonymize WSE query logs and their conclusion is that these techniques do not fit well with large sets of data that require a continuous update of the anonymization due to new data being added continuously.

Regarding the second case, an organization that builds query logs can work with "limited" sets of data by means of classifying received search queries by month, trimester or semester; then, at the end of the chosen period of time, just the corresponding and limited query log is anonymized and disclosed. This option clearly alleviates the cost required by the anonymization method in use; however, it only focus on a specific window of time, which implies that a lot of queries are discarded and, hence, the accuracy of the anonymized outputs is expected to be jeopardized due to the loss of information.

In conclusion, "closing" query logs, which are a dynamic source of data that is continuously growing with new search queries issued by the WSE's users, is contrary to their real nature and it has been acknowledged in the literature that it is an ineffective strategy. As a consequence, we need to study methods for real-time processing that may be applied to the considered scenario.

This approach, while it has not been directly applied to WSE query logs, is relevant for our research because it does not require to store a lot of data before being able to start any anonymization process. Schemes belonging

to this category consider data as a stream and they process information as soon as it is generated, introducing a minimum delay (Gaber et al., 2005). Data streams could be infinite and generate huge volumes of data. Therefore, mining massive data streams that have privacy protection in a network environment is a challenging task (Krempl et al., 2014).

Most approaches in the literature that try to address this field achieve k-anonymity and try to reduce the delay when processing data streams (Guo & Zhang, 2013; Zakerzadeh & Osborn, 2010; Yang et al., 2010; Li et al., 2008; Zhang et al., 2010; Zhou et al., 2009; Mortazavi & Jalili, 2014). These methods are mainly based on clustering incoming data streams and they all need to wait for new tupples in order to build the anonymized clusters. In order to avoid the problems inherent to accumulation-based methods, the authors of (Kim et al., 2014; Navarro-Arribas & Torra, 2014) follow alternative strategies that allow them to reduce delay.

Although the literature on this topic is rich, all the privacy-preserving systems designed to deal with data streams focus only on structured data. More specifically, the considered input data is mainly numerical, which is main limitation when considering its possible application to the anonymization of WSE query logs. Note that in the query logs of the WSEs it is usual to find unstructured data made of both textual and numerical data. This strong limitation shows that there is still room in the literature for new schemes specially designed to anonymize dynamic query logs while maximizing the data usability.

3. Our proposal

Our proposal, in a nutshell, is based on a server-side architecture that enables WSEs to anonymize query logs in a streaming environment. The outputs of this system are two: i) a real-time stream of anonymized logs; and ii) a database of user profiles. The main target is to allow WSEs to sell or release both data sets to interested third parties without threatening the privacy of the individuals who have filled the query logs with their issued search queries. In order to achieve that, the resulting outputs must fulfill certain requirements that are next detailed.

3.1. Requirements

3.1.1. Privacy requirements

The main requirement for the proposed system is that it must preserve the privacy of the individuals who contribute to the WSE query logs.

Privacy could be achieved by means of query de-identification, following one of the following approaches:

- *Full de-identification:* it is achieved when all identifiers, direct and indirect, are removed and there is no reasonable basis to believe that remaining information can be used to identify an individual.
- *Partial de-identification:* it is achieved when only direct identifiers are removed (indirect ones remain).
- *Statistical de-identification:* it is achieved by maintaining a trade-off between keeping as much useful data as possible while guaranteeing statistically acceptable data privacy.

Direct identifiers (e.g., full name, national id, etc.) can be easily removed from query logs. However, the textual content of search queries may contain any possible value of any domain; this is very likely to produce cases in which may be very difficult to distinguish identifying queries from innocuous ones. As a result, in the considered scenario full de-identification requirement cannot be guaranteed.

Partial de-identification is easier to achieve and, additionally, more data utility may also be preserved. However, it is prone to record-linkage attacks (Samarati & Sweeney, 1998) that would allow to re-identify users by means of certain apparently innocuous queries.

Statistical de-identification seems the more suitable approach. It is obtained trough certain statistical criteria and anonymization techniques, being k-anonymity and its extension l-diversity the two most widely accepted models. Those models were proposed for structured data (Zhang et al., 2010; Zhou et al., 2009; Torra & Domingo-Ferrer, 2001). However, the current proposal seeks to prove its usefulness also when anonymizing unstructured data streams.

3.1.2. Functional requirements

To enable the use of current proposal in a real environment, it must also fulfill a set of functional requirements:

- *Scalability:* It is the capability of a system to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth (Bondi, 2000). Specifically we want to achieve load scalability, that is the ability to easily expand or contract to accommodate heavier or lighter loads. Scalability methods fall into two broad categories (Michael et al., 2007): i) *Horizontal scalability* which is related to the ability of a system to add more working nodes, such as a new computer; and ii) *Vertical scalability* which is related to the ability of adding resources to a single node in a system, typically involving the addition of CPUs or memory. Both approaches have their own trade-offs, and the proposed architecture should take advantage of all the available resources. Configuring an existing idle system has always been the less expensive alternative, regardless of the approach. So the proposed architecture should be designed to take advantage of this fact and, therefore, it should use the existing WSE infrastructure.

- *Processing speed:* In order to minimize delays generating anonymized data stream, a high processing speed should be a main requirement. It's also fundamental in order to being able to process all requests received by the WSE. A WSE receives every second thousands of queries. For example, Google is processing in average 40000 queries per second (Internet Live Stats, 2016). This fact implies that the new proposal should be able to meet similar speed requirements as the ones met by WSEs.

- *Resource consumption:* As an additional requirement, the resource consumption of the system must be low, in order to facilitate its inclusion in an existing WSE, minimizing overhead cost. Even though the system is supposed to scale with added resources, it is important to not increase unreasonably current WSEs resource consumption.

- *Transparency:* To ease the integration of the proposed system in existing WSEs architectures, it must be transparent. New modules can hid internal details, making them invisible for the main architecture, i.e. encapsulated. Mainly, they should adhere to previous external interfaces without changing them, while changing its internal behavior, i.e. generating anonymized query logs with the same structure than original logs. The main purpose of providing a transparent system is to avoid changing any existing part of the WSE to be integrated with.

- *Modularity:* As explained above, the proposed solution should be easily integrable in an existing WSE. According to this, functional scalability should be achieved, i.e. being able to enhance the system by adding new functionality at minimal cost. To do achieve this, the proposed system should be designed to be modular and to have low coupling and high cohesion.

3.1.3. Utility requirements

Although full de-identification is desirable from a privacy point of view, ideally, preserving the privacy of the individuals should be compatible with allowing WSEs to sell non-sensitive user information to third parties. Logically, the economical value of these privacy-safe data will directly depend on its remaining quality from the utility point of view. As a result, there is a clear trade-off between anonymizing logs and keeping them useful to extract information through data mining processes. Therefore, the main challenge related to data utility is to anonymize sensitive user data, removing as few information as possible in order to have enough interesting information to be analyzed.

To achieve this, the proposed system aims to build fake logs and user profiles which should retain users' interests (maximizing the data utility) and eliminate any direct or quasi-identifier that could allow their re-identification (maximizing the user privacy). It should be as difficult as possible to relocate queries in order to build the original profile of a certain user.

3.2. Architecture

In Figure 1, three sub-systems that conform the proposed scheme are depicted. Note that, from those sub-systems, only the *WSE Anonymizer* should be integrated into the WSE environment. The other two are just defined to evaluate proposed method.

3.2.1. Actors

The main actors considered in the proposed process are the following:

- *WSE:* The web search engine. It builds the original query logs and it has the target of generating an anonymized version together with a users' profile database in order to sell or disclose them for economical purposes.

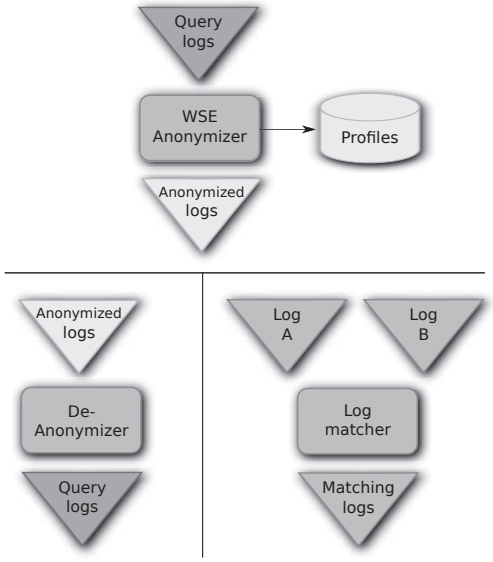

Figure 1: Main Architecture

- *Customer:* Represents a third party who wants to legitimately access the anonymized query logs and the users' profile database.
- *Attacker:* Represents a third party who wants to gain illegitimate access to the original query logs using as input the anonymized query logs and the users' profile database.

3.2.2. Phases

Current study is divided into the following phases:

- *Anonymization and profile creation*: Main phase, which takes the original query logs generated by the WSE as input and generates the anonymized query logs and a database of user profiles as outputs.
- *Anonymization analysis*: Anonymization and performance benchmarking, taking into account original data, anonymization time and resource usage.
- *De-anonymization*: Using the anonymized query logs as input, an attack is simulated, trying to link anonymized logs with the users that issued them.

- *Analysis*: De-Anonymization and performance benchmarking, taking into account original and anonymized data, time and resource usage.

3.2.3. Interactions

As the system is designed to anonymize a stream of query logs in real time, interactions are defined by production and consumption of streams of logs. The main interaction is defined between a WSE as producer, and a customer as consumer of the anonymized logs. An attacker will try to gain access to the anonymized stream of logs, such as any legitimate client and, then, it will try to de-anonymize them in order to recover the original query logs.

3.3. Proposal description

In this section all sub-systems that compose the architecture are described in more detail.

3.3.1. Anonymization and profile creation

In Figure 2, the main modules of the anonymization and profile creation sub-systems are represented. Each of them is responsible of a single action. All modules are described below.

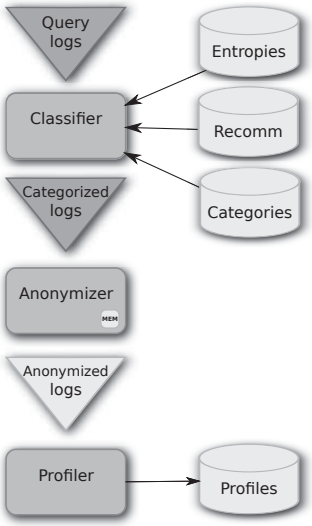

Figure 2: Anonymization and profile creation

Algorithm 1: Classifier

Input : query_logs, entropies, recommendations, categories
Output: categorized_logs

```
foreach log ∈ query_logs do
    SUs ← lemmatize(log);
    min_entropy ← ∞;
    foreach SU ∈ SUs do
        l ← recommendation(SU);
        e ← entropy(SU);
        if e < min_entropy then
            min_entropy ← e;
            main_SU ← l;
    end
    log_category ← category(main_SU);
    send log, log_category;
end
```

Classifier. Represented in Algorithm 1, the classifier uses query logs generated on the WSE as main input. It also needs access to a database of entropies, recommendations and categories for a given word. Once the classifier receives an user query, it's processed and categorized in real time. Then the categorized query is released. The process followed by the classifier is divided into the following stages:

- Natural language processing: Which is applied to query's text field to extract its semantic units (SUs) (Sánchez et al., 2013). Only nouns or adjectives are used. Then the HIPAA(U.S. Department of Health and Human Services, 1996) Privacy Rule is applied, to remove all the SUs that could be considered identifiers or quasi-identifiers, such as a name, address, phone number etc.

- Recommendation: To correct possible misspellings, selected SUs are validated against a recommendation database, which returns the correct term in case of a misspelled word.

- Entropies: The amount of information (entropies) are calculated for each of the remaining SUs. To do so, a database of entropies is used,

which should provide the number of references on the WSE for a given term. The algorithm will chose as *main_SU* the one which has less references, which is assumed to be the most specific and, hence, informative term of the query (Sánchez et al., 2010).

- Categorization: The corresponding category for the query is then calculated using *main_SU* and a database of categories, obtaining the most specific topic of the query.

As a representative example of how the *Classifier* works, let us assume an input query whose keywords are: "european soccer barcelona players". The first step of this process extracts the SUs: "european soccer", "barcelona" and "players". Next, entropies are calculated for each SU, getting the following results according to Google's WSE: "european soccer": 56,500,000 results; "barcelona": 504,000,000 results; and "players": 544,000,000 results. According to these numbers, "european soccer" is selected as the *main_SU* of the input query. In the last step, "european soccer" (as *main_SU*) is used to assign a corresponding category of interest to the input query by means of a knowledge base. In this way, if the *Open Directory Project (ODP)* [2] is used as knowledge base and queried with "european soccer", retrieving "Sports" as the resulting category. Therefore, the input query is categorized as related to "Sports".

Anonymizer. Represented in Algorithm 2, it uses as input the categorized logs that are generated in real time by the classifier (see the previous explanation for more details about this). Those logs are split in two sets for each category, one that stores the users, and another that stores the text of the queries. When a category set reaches the maximum allowed value, defined by k, the algorithm randomly takes an user and a query from that category and builds with those the anonymized query with a minimum delay.

A special case occurs when all users stored in certain category set are the same. In this case, the selected user would be the same than the original one. In order to prevent this situation (which, in any case, it is very unlikely to happen due to the huge quantity of queries that a WSE receives each second), we impose an additional restriction, not shown in Algorithm 2 for simplicity:

[2] Open Directory Project. `http://www.dmoz.org/`

Algorithm 2: Anonymizer

Input : categorized_logs, k, δ
Output: Anonymized logs

```
foreach log ∈ categorized_logs do
    user, query, category ← log;
    users[category] ← user;
    query[category] ← query;
    if size(users[category]) = k then
        if ∀u ∈ users[category], ∃u ≠ user then
            pop random u ∈ users[category];
            pop random q ∈ query[category];
            send u, q;
        else k = k * δ;
end
```

When all users in a certain category are the same, the anonymizer must increment the corresponding k value. This is done by multiplying k by a δ value. With this step, we ensure that the result is also l-diverse.

As an example of how the proposed *Anonymizer* works, let us assume that the system works with parameter $k = 4$ and that in the data structure in charge of storing queries related to "Sports" there are already stored three queries: i) "novak djokovic tennis titles", sent by user A; ii) "monza formula1 tickets", sent by user B; and iii) "champions league final", sent by user C. The *Anonymizer* receives the query "european soccer barcelona players" that has been categorized as "Sports" in the last phase by the *Classifier*. The data structure "Sports" contains now four queries so, it is full, according to k; therefore, one of the stored queries is selected at random and it is assigned to a sender at random too. Following this example, let us assume that "champions league final" and user A are randomly selected. As a result, an anonymized log is outputted where user A is now linked to "champions league final" instead of to her original query "novak djokovic tennis titles". It can be seen that, by doing this, the generality is kept (i.e., A is interested in "Sports") while the specificity is eliminated (i.e., A was specifically interested in "Tennis" instead of "Soccer"). As explained in works such as (Viejo & Sánchez, 2014), keeping general interests improves the utility of the anonymized data while hiding specific interests is useful to preserve the

privacy of the respondents.

Profiler. This element uses as input the anonymized record generated by the anonymizer and the corresponding category. With this data a user profile is created or updated in real-time. Therefore, the profiler is in charge of keeping the profile database updated. As it has been explained in the Introduction, in the literature, a user profile is generally considered a set of well-defined categories of interests (e.g., science, health, society, sports, etc.) with a certain weight assigned to each one according to the evidences generated by the corresponding user and how they have been classified under each category (Viejo et al., 2012).

As an example of how the proposed *Profiler* works, let us assume that the system uses the following set of categories: Arts, Health, Shopping, Science, Computers, Sports, Society and Business (note that the proposed system is not bounded to this set of categories, this is only an example). Let us assume also that a certain individual has already sent: 5 queries related to Science; 10 queries related to Business; 20 queries related to Arts; 5 queries related to Health and 10 queries related to Computers. As a result, the current profile for that person stored in the profile database is: (Arts: 40%, Health: 10%, Shopping: 0%, Science: 10%, Computers: 20%, Sports: 0%, Society: 0%, Business: 20%). Now, let us assume that this person is assigned to the new query "champions league final" by the *Anonymizer*, this query has been categorized as "Sports" by the *Classifier*. The *Profiler* obtains the new evidence and updates the corresponding user profile as follows: (Arts: 39.2%, Health: 9.8%, Shopping: 0%, Science: 9.8%, Computers: 19.6%, Sports: 1.9%, Society: 0%, Business: 19.6%).

3.3.2. De-Anonymization

The purpose of this process is to try to link the anonymized logs with the individuals who generated them. By this way, the system tries to evaluate whether the anonymization process executed previously has been successful or not and, therefore, whether the resulting protected query logs can be disclosed or not.

As seen in Figure 3, the de-anonymization process is very similar to the anonymization one. We assume that the attacker has already gained access to the stream of anonymized logs generated by the WSE. Those logs are the main input for the process.

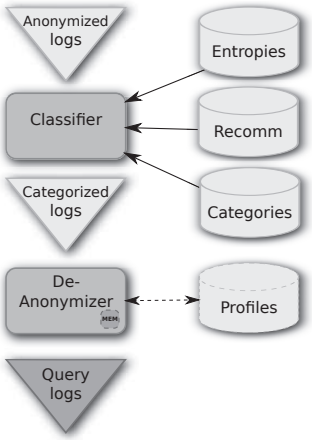

Figure 3: De-Anonymization

First of all, the input logs should be classified by the attacker. Best de-anonymization could be achieved if the attacker manages to get the same categorization used by the WSE (this is also the worst-case scenario form a privacy point of view). As result, the attacker can use Algorithm 1 to categorize the anonymized logs prior to perform their de-anonymization. In this way, the attacker also needs access to a database of entropies, recommendations and categories.

Then, each categorized log is send to the de-anonymizer algorithm, which will try to recover the original query logs. Several de-anonymizer algorithms are proposed and tested on next section. All of them share the same basic structure, shown on Algorithm 3.

The de-anonymization algorithm also uses two additional parameters, k and δ. Their function is the same than in Algorithm 2. An attacker will always attempt to use similar values to the ones used in the anonymization process.

Each de-anonymizer variation defines a different *pop_selected* function, and some specific data structures, represented with dashed lines on Figure 3. All those variations are detailed in Section 4.

Following the same example used to explain the work of the *Anonymizer*, in this case, the *De-anonymizer* takes as input the anonymized query logs. Let us assume that this is: i) "champions league final", sent by A; ii) "monza formula1 tickets", sent by Z; iii) "novak djokovic tennis titles", sent by B; and iv) "european soccer barcelona players", sent by C. The *De-anonymizer* first

Algorithm 3: De-Anonymizer

Input : categorized_logs, k, δ
Output: query_logs

```
1  foreach log ∈ categorized_logs do
2      user, query, category ← log;
3      users[category] ← user;
4      query[category] ← query;
5      if size(users[category]) = k then
6          if ∀u ∈ users[category], ∃u ≠ user then
7              pop_selected u ∈ users[category];
8              pop_selected q ∈ query[category];
9              send u, q;
10         else k = k * δ;
11 end
```

categorizes the anonymized query logs to ascertain the category of interest linked to each query (as done in the anonymization process) and then tries to link a certain query with its original sender. The process followed to do that may vary and different approaches are explained in the Implementation section. A simple approach would be to match sender and query keywords at random (as done in the anonymization process). Using this method in this example may lead to match B with "european soccer barcelona players" which would get a failure in the de-anonymization process, or with "monza formula1 tickets" which would get a success.

3.3.3. Analysis

Finally we also need a basic algorithm to verify that the proposed scheme is working properly, this is a log matcher. More concretely, it gets two log streams as input, and returns the number of matching logs, i.e. identical logs appearing in the two input streams. A resource profiler is also needed for a proper analysis, which at least should calculate amount of time and resources used in each task.

4. Evaluation

In this section, the implementation used to test our proposal is described. All the conducted tests are also detailed. Finally a discussion of system

performance in terms of user's privacy, functional requirements and data utility is provided.

4.1. Implementation

All algorithms described in Section 3 were implemented in *Python*. Input and output query logs are stored in plain text files preserving the original format of logs. Communication between modules was also done via plain text files, to enable posterior analysis, but all modules could also use a sockets based communication. A *No-SQL* database, was used to store user profiles as well as other persistent data. Beside those implementation decisions, all other major changes that have been made to the initial algorithms during the implementation process, are discussed below.

Classification is a complex task to achieve outside of a WSE context, mainly because, besides the original logs, some additional information is required. As previously described, the proposed algorithm needs a database of word entropies, a database of word recommendations and a database of word categories. In a WSE, this information would be provided by the WSE itself. It should be noted that a WSE already generates some of this information with each search, therefore it would not represent an extra cost. Outside of a WSE, this information should be retrieved querying an external WSE during the classification process but, by doing this, the obtained results in terms of time and resource usage may differ significantly from the ones that would be achieved in a real environment. In order to prevent this situation, a web scrapper was implemented as a preparatory step. The web scrapper retrieves information from remote sources and creates three local *No-SQL* databases that are the ones used in the classification step, therefore obtaining an environment similar to a real one.

Once classifying the data, we also need a way to lemmatize each query text, which is a phrase written on natural language. As the system was implemented in *Python*, the *NLTK* package (Natural Language Toolkit) [3]. This package provides interfaces to corpora and lexical resources such as *WordNet*, along with text processing libraries. On preliminary tests, *NLTK* was very slow for the requisites of our system, but due to the fact that it is a very generic system prepared for a wide variety of texts, uses and situations, a modified version of *NLTK* designed to fulfill our requirements was developed.

[3]Natural language toolkit. `http://www.nltk.org/`

Table 1: AOL query log example

1887264 5424618 ninja turtles rap lyrics 2006-05-22 17:04:54 1 http://www.lyrics007.com
1887267 5424618 myspace.com 2006-05-22 17:55:28 2 http://music.myspace.com
1887549 5426574 rio hotel and casino 2006-03-20 18:36:01 5 http://www.destination360.com
1887552 5426574 orleans hotel casino 2006-03-21 16:06:02 3 http://www.tickco.com
2536798 9146863 invest in spring drinking water 2006-03-07 13:51:21 2 http://www.fool.com
2536814 9146863 spring water stocks to buy 2006-03-13 22:13:44 4 http://importer.alibaba.com

This modified *NLTK* showed the same utility as the original package when dealing with query logs but it became hundreds orders of magnitude faster due to their focus on a more specific duty.

4.2. Evaluation methodology

In order to evaluate the architecture proposed in Section 3 we have implemented our system as described in Section 4.1. Only those systems which would be used on a real environment are evaluated regarding privacy, functional and utility requirements.

4.2.1. Data

We ran our experiments on logs released by *AOL* [4] stored on plain text files, and on a database of word entropies, a database of word recommendations and a database of word categories, stored in a NoSQL database. Released *AOL* data contains 36389576 query logs, corresponding to a period of three months of real activity. Table 1 provides a brief sample of the used logs.

Databases of word entropies, recommendations and categories are created using the web scrapper defined in Subsection 4.1. Since the database content is generated based on logs content, databases only contain log related data, which can slightly affect system performance measures. A first attempt of data gathering was conducted using Google but it applies a very restrictive limit to the number of search queries from a certain source that can be served; as a result, our processes were blocked. Finally, *Microsoft Bing* was used to fill word entropies and recommendations databases. To create database of word categories, *Open Directory Project (ODP)* was used. ODP is a large, categorized directory of websites and pages, which is managed by volunteers.

Once the databases were created, no other query was done to any WSE, and all tests were conducted using those databases as the only data repos-

[4] AOL keyword searches. `http://www.gregsadetsky.com/aol-data/`

itory. At the end of scrapping process, databases contained 1587451 word categorizations, 1751632 word entropies and 258504 word recommendations. Note that some query logs contain a query text with no understandable value. Grammatical mistakes were resolved using a word recommendation database. Other queries contain no query text, or it does not make any sense, being just lots of consonants concatenated. As a category for those last logs cannot be found, they were discarded.

4.2.2. Conducted tests

Only two parameters can be modified in the proposed system: k and δ. Therefore, several test were conducted to determine the effects of different k and δ values. It was also necessary some additional testing to determine the accomplishment for the rest of the requirements defined on Section 3.

δ adjustments. Some preliminary test were conducted to determine the effect of δ value. As explained on the algorithm definition, δ determines how fast k value increases when all k-elements on a category are the same. With a small δ the category size should be increased more times until we get different elements in the category and, therefore, l-diversity. But with a small δ, the final k value fits best the needs of the data. With a large δ, category size should be increased less times, but the final k value should be bigger than the needs of the data.

Figure 4 shows that the size of δ does not affect the running time of the proposed system, hence, this is not a relevant factor to fix a certain δ value. Regarding the effect of δ on k value during the execution of the anonymization process, Figure 5 depicts that, even though bigger δ values produce slightly bigger final k values reached by the system after multiple iterations, the difference obtained is not significant enough. Due to the fact that a small δ provides a more accurate final k value and smaller memory consumption without affecting other parameters, we decided to fix δ to 1.2 for the following tests. This leaves k as the only adjustable parameter of the proposal. As stated on following sections, the system performance, as well as the user privacy, depend on k.

Classifier. The proposed classifier cannot be customized in any way; therefore, only some functional and utility tests were conducted.

Anonymizer. Privacy, functional and utility tests were conducted for the anonymizer, all of them with various values of k.

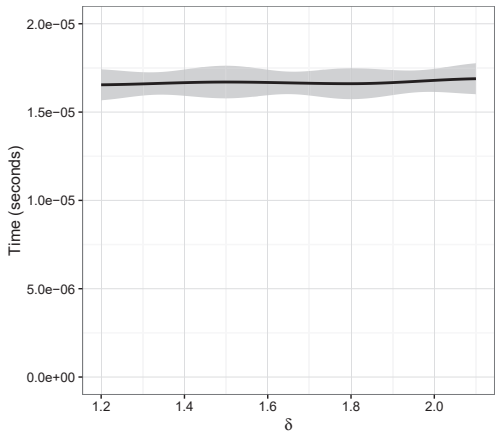

Figure 4: Effects of δ on time

Profiler. The proposed profiler cannot be customized in any way; therefore, only some functional and utility tests were conducted.

De-anonymizer. In order to perform detailed privacy tests, some attacks were simulated using different variations of the de-anonymizer algorithm. In all these cases, the time field contained in the logs was used to consider the proper order in which each search query was received. We next detail each variation of the de-anonymizer element:

- *De-anonymizer 1*: This approach tries to retrieve original logs choosing one random log and one random user from the k-element sets, which the de-anonymizer recreates, conducting the same operations than the WSE.
- *De-anonymizer 2*: This approach does the same than the first version, but instead of selecting a random user, selects the user who appears more times in the k-element set linked to a certain category.
- *De-anonymizer 3*: This approach has access to the number of queries related to each category that were sent by this user until that moment. In this way, from the users who appear in the k-element set linked to a certain category on a given time, the proposed algorithm selects the user who has sent more queries related to this category. Therefore, this method makes its decision taking into account the query history of the

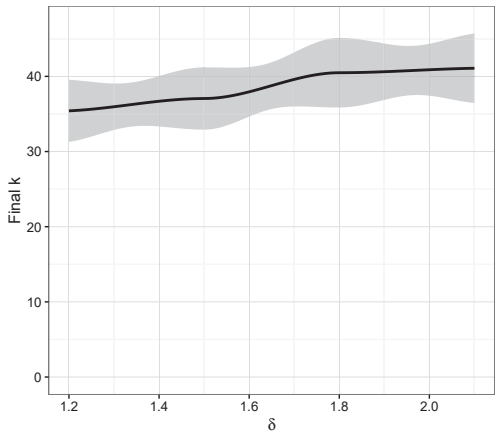

Figure 5: Effects of δ on final k

respondents instead of just their temporal appearance in the k-element set linked to a certain category of interest.

- *De-anonymizer 4*: Uses the same approach than in the third version, but the number of queries related to each category that are obtained from the query history are multiplied by the number of times that the respondent appears in the k-element set linked to a certain category. The respondent with the highest result is then assigned to the current query. This approach considers the query history but tries to give more weight to the fact that an individual has sent a certain type of queries recently (which are those stored in the k-element set).

4.2.3. Test environment

All experiments were performed using a *Dell* notebook running *Ubuntu Linux* 14.04 LTS, with a 1.8 GHz *Intel Core*$^{\text{TM}}$i7-4500U CPU and 8GB of RAM. System hard disk was a *Seagate ST1000LM014*, which performance profile is skewed strongly towards small file I/O, and a below average overall performance. Much better results could be obtained with a faster hard disk or *Solid State Disks* (SSD), that are already installed in many servers nowadays. All algorithms were implemented and executed in *Python* 2.7.6. *MongoDB* 3.0.7 was used as *No-SQL* database, which was also installed and running on the same computer.

4.3. Privacy study

To test the privacy level provided by the new proposal, the original query logs were compared to the anonymized query logs, counting the percentage of matching records. Results can be seen on Figure 6.

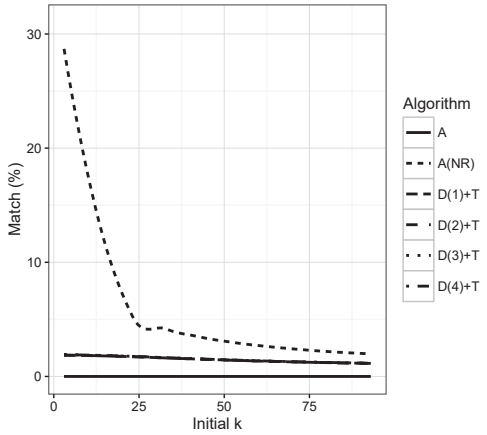

Figure 6: Matching records

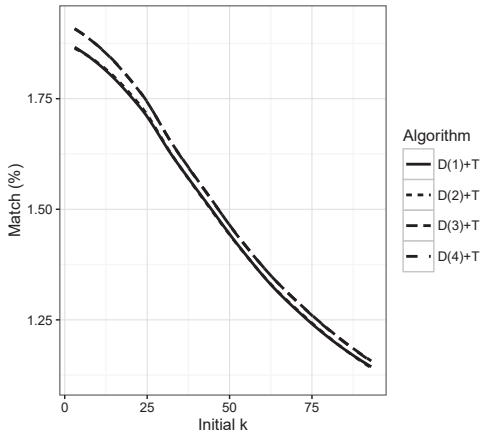

Figure 7: Matching records. De-anonymizer

In this figure, the case *A(NR)* represents what would happen if the anonymizer does not increment the corresponding k value when all users stored in certain category set are the same (this is a special case explained in Section 3.3.1). In this case, the figure depicts that k remains constant

but the output contains non-anonymized logs. When this special case is covered by means of incrementing k, better privacy is obtained. Data generated by the anonymizer, A in the figure, do not contain any matching registers; therefore, full privacy seems to be obtained.

However, all de-anonymizer algorithms described above were used with the anonymized logs in order to try to reconstruct the original log stream. Results can be seen in Figure 6 and, more concretely, Figure 7 focuses on the four proposed de-anonymizers. The $+T$ element means that those de-anonymizers use logs sorted according to time field. In all the tests, all the de-anonymizer versions performed in a similar way (some differences applies but they are not significant) and they only were able to recover around 1.89% of original logs with the lowest k value (best case for the de-anonymizers). As a result of our tests, it can be concluded that it is quite difficult to link the anonymized logs with their original users.

4.4. Functional study

We next discuss the level of achievement of the functional requirements defined previously.

4.4.1. Modularity

The proposed system was developed as a set of independent services, forming a micro-service architecture. It allows to accomplish a low coupling and high cohesion system. This architecture makes the system more reliable and easy to maintain. Some services could be changed by existing modules in the WSE, e.g. a WSE classifier. Other services could optionally be deployed or not depending on the needs of the WSE. For example, the profiler should be used if the WSE is willing to create anonymous user profiles but, if the WSE only desires to release an anonymized log stream, the proposed system would also work without it.

4.4.2. Scalability

Thanks to the high modularity of the developed services, it is easy to deploy them on available idle systems, using the existing WSE infrastructure. Horizontal scalability could be achieved by deploying more than one instance of each service, either at the same or at different systems. Vertical scalability could be also achieved, since a faster CPU, disk or some memory dedicated to cache data would increase significantly the amount of work the system could handle. In lighter loads, modules will remain idle, consuming an insignificant

amount of resources. Most modules could be closed completely if not used, for example, the classifier and the profiler. So this system shows a high load scalability.

4.4.3. Speed

Time consumption of WSE processes can be seen in Table 2. Even though the computer used for the tests has 4 cores, only 1 was used in each test. All algorithms support multi-threading, so better results could be obtained by simply enabling it.

Table 2: Runtime cost

	Time/log (μs) Mean $\pm$ SD	Queries/second (Threads Google)
Classifier	1503 $\pm$ 24.0	665 (60)
Anonymizer	22 $\pm$ 0.35	45454 (1)
Profiler	267 $\pm$ 4.73	3745 (11)

The classifier and the profiler do not use k values and, hence, they are not affected by its variations. In the other hand, the anonymizer uses k, but results do not change significantly with different k values, as it is reflected by the small standard deviation.

The classifier was the slowest algorithm, because it has to search in several databases. We must be aware of two factors that affect its performance: i) the content of those databases was mainly generated using logs' content, therefore, databases only contain log related data; and ii) the majority of the time used by the classifier corresponds to I/O from disk, hence, a faster disk, or placing the data in the system's memory would greatly improve the performance of the classifier. The same idea could be applied to the profiler, which creates users profiles on a disk database.

Using as reference the 40000 queries per second that Google processes on average, one single thread of the anonymizer in the used test environment can anonymize all Google queries in real-time. On the other hand, we need 60 classifier threads and 11 profiler threads to reach real-time performance in our test environment, but with the discussed changes, those numbers could be lower on a real setting.

For completeness, all de-anonymizer algorithms were also tested. Results can be seen in Figure 8. As expected, algorithms that count the number of times that a user appears on a category, are the slowest overall, and more

affected when k increases. Using profiles and choosing the best fit on a category was also affected by k.

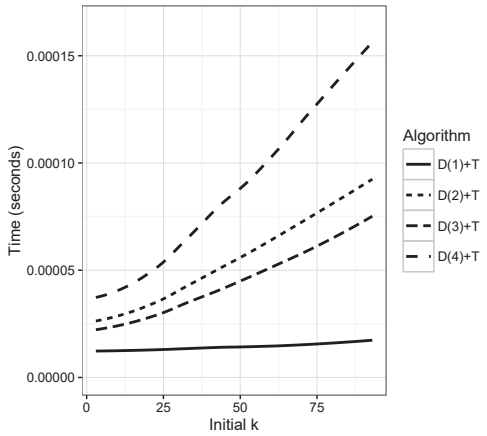

Figure 8: De-Anonymizer speed

4.4.4. Resource consumption

Table 3 shows the achieved results for maximum memory and disk usage at each process. It's important to note that de-anonymizer-3 and de-anonymizer-4 keep basic user profiles in memory, and that is the reason because they consume more memory. Apart from those two algorithms, the rest should not use more memory, whatever the amount of logs they deal with.

Regarding the disk space, when output log streams are stored in the disk, they use exactly the same amount of space than the original log streams. Furthermore, if the profiler stores queries in a user profile, with 4GiB of test data, it generated a 3GiB profiles database, with an average record size of 112 bytes. Note that, in any case, this will depend on the input data.

4.4.5. Transparency

This proposal only needs to read the stream of logs already generated by a WSE, so it is no necessary to make changes at the existing WSE architecture. The resulting anonymized log stream has exactly the same structure and size than the original one. Therefore, this system can be plugged at the end of the current process followed by the WSE and generate an anonymized output without changing anything else.

Table 3: Memory usage

	Max. mem (KiB) Mean ± SD
Classifier	111581 ± 145
Anonymizer	10347 ± 72
Profiler	12120 ± 94
De-anonymizer 1	9316 ± 640
De-anonymizer 2	9375 ± 687
De-anonymizer 3	312273 ± 4620
De-anonymizer 4	311782 ± 2945

4.5. Utility study

This section studies the classifier and the anonymizer in terms of utility.

Even though the classifier obtains a hierarchical categorization for each query, the anonymizer sets are based only on very generic categories. When a query cannot be classified into any category, the classifier uses Microsoft Bing recommendations to find alternatives. As a result, the proposed classifier was capable of categorizing 85% of the all the tested queries. In order to verify that this automatic categorization was working properly, a sample of 1068 logs were manually classified and compared to the results of the algorithm. This sample, in our population of 36389576 logs, provides a confidence level of 95%, with a margin of error of 3%. Through this comparison we found 58.99% of logs in a correct category. In Table 4 all used categories can be seen. Those categories correspond to the first hierarchical level of ODP. Percentage of correctly classified logs for each category was also estimated.

It can be seen that this strategy works better in some categories. For instance, with categories that contain very specific terms, such as *Health*, the best results are obtained. In contrast, worse results are gathered in categories with generic terms, such as *News*. This is because the proposed algorithm only selects the word with lower entropy, which is expected to be the most specific one. Categorization was solely based on this word. When a combination of words changes the meaning of the query, this approach led to errors. For example, in the query "artic monkeys", the lowest entropy word is "artic". This word is categorized as "Regional: Polar Regions". That is correct for the word itself but using the whole context, category should be "Arts: Music". These results show that a better classifier should only be obtained by means of applying more complex natural language processing

Table 4: Classification utility

	Classification % Correct		Classification % Correct
Arts	57.23%	Recreation	55.22%
Business	77.39%	Reference	69.23%
Computers	64.79%	Regional	69.08%
Games	56.25%	Science	47.22%
Health	88.10%	Shopping	69.64%
Home	81.48%	Society	56.76%
Kids and Teens	37.65%	Sports	40.48%
News	16.67%	World	30.77%

systems; however, this is outside of the scope of this proposal.

In addition to that, to check the anonymizer's utility, two sets of user profiles were created using the profiler. The first set contained profiles created with original logs while the second set contained profiles created with anonymized logs. Both sets were compared in order to check whether they were equivalent. It was confirmed that, for a given user, both sets of profiles contained the same number of queries for each category. We argue that if an anonymized profile contains the same categories with the same weights than an original profile, it implies that the anonymized profile retains the same utility than the original one.

5. Conclusions and further research

This paper has presented a novel framework for anonymizing query logs generated by WSEs. Privacy guarantee is defined in terms of set theory, which relates sets of users to sets of query logs. Data could be released without other modifications than removing direct identifiers from query text and remapping between those two sets. This contrasts to existing approaches that release heavily modified data, either distorted or generalized, to maintain anonymity.

In our evaluation, we have considered the worst-case scenario, in which an attacker who is willing to use the anonymized query logs to retrieve the original query logs has gained access to the same base information and algorithms than the WSE. We conducted tests under this context and the best attempt to recover the original logs only obtained a 1.89% of them. All the

proposed methods were tested using the AOL released logs; therefore, we argue that our solution is capable of dealing with real data in a real setup. In this way, we have considered also the average Google's load to study the runtime cost and the memory usage. The query log's utility after its anonymization was also analyzed, and the results showed that original query logs and anonymized query logs were equivalent in terms of categories being reflected.

Regarding future work, the proposed classifier may be improved by means of a more accurate natural language analysis in order to perform a semantic analysis of queries. Regarding the runtime of the proposed algorithms, the classifier was also the slowest and, therefore, there is room to develop more efficient alternatives. Note that having a better categorization system should also provide a better level of utility.

Disclaimer and acknowledgments

This work was partially supported by the Spanish Government under CO--PRIVACY TIN2011-27076-C03-01, ARES-CONSOLIDER INGENIO 2010 CSD 2007-00004 and BallotNext IPT-2012-0603-430000 projects. Authors are members of the UNESCO Chair in Data Privacy, yet the views expressed in this paper neither necessarily reflect the position of the UNESCO nor commit with that organization.

References

Adar, E. (2007). User 4xxxxx9: Anonymizing query logs. In *Proc. of the 16th International World Wide Web Conference – WWW'07*.

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *Proc. of the 29th annual ACM SIGIR conference*.

Bar-Ilan, J. (2007). Access to query logs an academic researchers point of view. In *Proc. of the 16th International World Wide Web Conference – WWW '07*.

Barbaro, M., & Zeller, T. (2006). A face is exposed for aol searcher no. 4417749. In *The New York Times*.

Batet, M., Erola-Ferrer, A., Sánchez, D., & Castellà-Roca, J. (2013). Utility preserving query log anonymization via semantic microaggregation. *Information Sciences*, *242*, 49–63.

Beitzel, S. M., Jensen, E. C., Chowdhury, A., Grossman, D., & Frieder, O. (2004). Hourly analysis of a very large topically categorized web query log. In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval – SIGIR'04*.

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Proc. of the 2nd International Workshop on Software and Performance*.

Brenes, D. J., & Gayo-Avello, D. (2009). Stratified analysis of aol query log. *Information Sciences*, *179*, 1844–1858.

Center for Democracy and Technology (2007). Search privacy practices: A work in progress.

Chau, M., Fang, X., & Liu Sheng, O. R. (2005). Analysis of the query logs of a web site search engine. *Journal of the American Society for Information Science and Technology*, *56*, 1363–1376.

Cooper, A. (2008). A survey of query log privacy-enhancing techniques from a policy perspective. *ACM Transactions on the Web*, *2*, 19:1–19:27.

Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams: a review. *ACM SIGMOD Record*, *34*, 18–26.

Guo, K., & Zhang, Q. (2013). Fast clustering-based anonymization approaches with time constraints for data streams. *Knowledge-Based Systems*, *46*, 95–108.

He, Y., & Naughton, J. F. (2009). Anonymization of set-valued data via top-down, local generalization. In *Proc. of the Proc. of the VLDB Endowment*.

Hong, Y., He, X., Vaidya, J., Adam, N., & Atluri, V. (2009). Effective anonymization of query logs. In *Proc. of the 18th ACM Conference on Information and Knowledge Management – CIKM09*.

Internet Live Stats (2016). Google search statistics. http://www.internetlivestats.com.

Jansen, B. J. (2006). Search log analysis: What is it; what's been done; how to do it. *Library and Information Science Research*, *28*, 407–432.

Jones, R., Kumar, R., Pang, B., & Tomkins, A. (2007). I know what you did last summer. In *Proc. of the 16th ACM conference on Conference on information and knowledge management – CIKM'07*.

Jones, R., Rey, B., Madani, O., & Greiner, W. (2006). Generating query substitutions. In *Proc. of the 15th international conference on World Wide Web – WWW'06*.

Kim, S., Sung, M. K., & Chung, Y. D. (2014). A framework to preserve the privacy of electronic health data streams. *Journal of Biomedical Informatics*, *50*, 95–110.

Korolova, A., Kenthapadi, K., Mishra, N., & Ntoulas, A. (2009). Releasing search queries and clicks privately. In *Proc. of the 18th International World Wide Web Conference – WWW '09*.

Krempl, G., Zliobaite, I., Brzezinski, D., Hullermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., & Stefanowski, J. (2014). Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, *16*, 1–10.

Li, J., Ooi, B. C., & Wang, W. (2008). Anonymizing streaming data for privacy protection. In *Proc. of the IEEE 24th International Conference on Data Engineering*.

Michael, M., Moreira, J. E., Shiloach, D., & Wisniewski, R. W. (2007). Scale-up x scale-out: A case study using nutch/lucene. In *Proc. of the 2007 IEEE International Parallel & Distributed Processing Symposium –IPDPS'07*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *38*, 39–41.

Mills, E. (2006). Aol sued over web search data release. In *CNET*.

Mortazavi, R., & Jalili, S. (2014). Fast data-oriented microaggregation algorithm for large numerical datasets. *Knowledge-Based Systems*, *67*, 195–205.

Navarro-Arribas, G., & Torra, V. (2009). Tree-based microaggregation for the anonymization of search logs. In *Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*.

Navarro-Arribas, G., & Torra, V. (2014). Rank swapping for stream data. In *Proc. of the 11th International Conference on Modeling Decisions for Artificial Intelligence – MDAI'14*.

Netcraft (2016). February 2016 web server survey. http://news.netcraft.com/.

Poblete, B., Spiliopoulou, M., & Baeza-Yates, R. A. (2007). Website privacy preservation for query log publishing. In *Proc. of the 1st ACM SIGKDD international conference on Privacy, security, and trust in KDD – PinKDD'07*.

Richardson, M. (2008). Learning about the world through long-term query logs. *ACM Transactions on the Web*, *2*.

Saagar, K. (2014). Monetizing data: Milking the new cash cow. In *Wired*.

Samarati, P., & Sweeney, L. (1998). *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*. Technical Report SRI International.

Sánchez, D., Batet, M., Valls, A., & Gibert, K. (2010). Ontology-driven web-based semantic similarity. *Journal of Intelligent Information Systems*, *35*, 383–413.

Sánchez, D., Castellà-Roca, J., & Viejo, A. (2013). Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines. *Information Sciences*, *218*, 17–30.

Shen, X., Tan, B., & Zhai, C. X. (2007). Privacy protection in personalized search. *ACM SIGIR Forum*, *41*, 4–17.

Soghoian, C. (2007). The problem of anonymous vanity searches. In *Social Science Research Network*.

Soria-Comas, J., & Domingo-Ferrer, J. (2015). Big data privacy: challenges to privacy principles and models. *Data Science and Engineering*, *1*, 1–8.

Tancer, B. (2008). *Click: What Millions of People Are Doing Online and Why it Matters*. Hyperion.

Terrovitis, M., Mamoulis, N., & Kalnis, P. (2008). Privacy-preserving anonymization of set-valued data. In *Proc. of the Proc. of the VLDB Endowment*.

Torra, V., & Domingo-Ferrer, J. (2001). Disclosure control methods and information loss for microdata. In *Confidentiality, disclosure, and data access: Theory and practical applications for statistical agencies*. Elsevier.

U.S. Department of Health and Human Services (1996). Health insurance portability and accountability act. http://www.hhs.gov/hipaa.

U.S. Federal Trade Commission (2014). Data brokers, a call for transparency and accountability.

Viejo, A., & Sánchez, D. (2014). Profiling social networks to provide useful and privacy-preserving web searchs. *Journal of the Association for Information Science and Technology*, *65*, 2444–2458.

Viejo, A., Sánchez, D., & Castellà-Roca, J. (2012). Preventing automatic user profiling in web 2.0 applications. *Knowledge-Based Systems*, *36*, 191–205.

Yang, G., Yang, J., & Chu, Y. (2010). Research on data streams publishing of privacy preserving. In *Proc. of the 2010 IEEE International Conference on Information Theory and Information Security –ICITIS'10*.

Zakerzadeh, H., & Osborn, S. L. (2010). Faanst: fast anonymizing algorithm for numerical streaming data. In *Proc. of the 5th international Workshop on data privacy management – DPM10*.

Zhang, J., Yang, J. C., Zhang, J., & Yuan, Y. (2010). Kids:k-anonymization data stream base on sliding window. In *Proc. of the 2010 2nd International Conference on Future Computer and Communication*.

Zhou, B., Han, Y., Pei, J., Jiang, B., Tao, Y., & Jia, Y. (2009). Continuous privacy preserving publishing of data streams. In *Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology – EDBT'09*.