Francisco Javier Rodrigo Ginés

DEVELOPMENT OF A PRIVACY TOOL FOR THE PROTECTION OF PROFILES IN SEARCH ENGINES

FINAL MASTER'S PROJECT

Directed by Dr. Javier Parra Arnau

Master's Degree in Computer Security and Artificial Intelligence

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2018

UNIVERSITAT ROVIRA I VIRGILI

Abstract

Escola Técnica Superior d'Enginyeria

Department of Computer Science and Mathematics

Web search engines capitalize on, or lend themselves to, the construction of user interest profiles to provide personalized search results. The lack of transparency about what information is stored, how it is used and with whom it is shared, limits the perception of privacy that users have about the search service. In this thesis, we investigate a technology that allows users to replace specific queries with more general but semantically similar search terms.

Through the generalization of queries, the user profile becomes less precise and therefore more private, although evidently at the expense of a degradation in the accuracy of the search results. In this work, we design and develop a tool that implements this principle in real practice. Our tool, developed as a browser plug-in for Google Chrome, enables users to generalize the queries sent to a search engine in an automated fashion, without the need for any kind of infrastructure or external databases, and in real time, according to simple and intuitive privacy criteria.

Experimental results demonstrate the technical feasibility and suitability of our solution.

Index

1. INTRODUCTION

Billions of queries are processed daily by Web search engines (WSEs) such as Google, Bing or Yahoo. Naturally, users of these services look for information that is relevant to their interests, and this is the fundamental reason why WSEs strive to develop increasingly sophisticated algorithms that tailor search results to meet the specific preferences of their users.

Personalization, the key enabling technology, relies on the storage of user information (e.g., the queries themselves, the search results visited, the location from which queries are submitted), the processing of such data, and the creation of a profile of interests and preferences. With this profile, search engines can then adjust search results (Bin Cao, 2017), (Elif Aktolga, 2013) to provide users with more accurate links.

Fig. 1. The profile of a user is modelled in Google as a list of topic categories. The profile shown here reflects the user is interested in parenting-related topics, which might reveal she is pregnant.

Behind personalization, however, WSEs not only aim to offer more precise search results —the construction of profiles allows search engines to segment their users and deliver personalized ads, which have been shown to ensure

conversion rates[1] that double those of geographical and contextual ads (Beales, 2010). Internet companies, besides, very often obtain a direct economic benefit through the sale of this valuable information. This is the case of Yahoo, which claims to charge the US government between 30 and 40 dollars for the email address of users of its search engine (Zetter, 2009).

Evidently, personalization techniques —and in particular the creation of interest profiles—, prompts serious privacy risks. On the one hand, the lack of transparency about what information is stored, how it is used and with whom it is shared, limits the perception of privacy that users have about the search service. On the other hand, user's profiles are sensitive information per se since they may reveal health-related issues, political affiliation, salary or religion. Fig. 1 shows an example of user profile and the inferences that can be drawn from it.

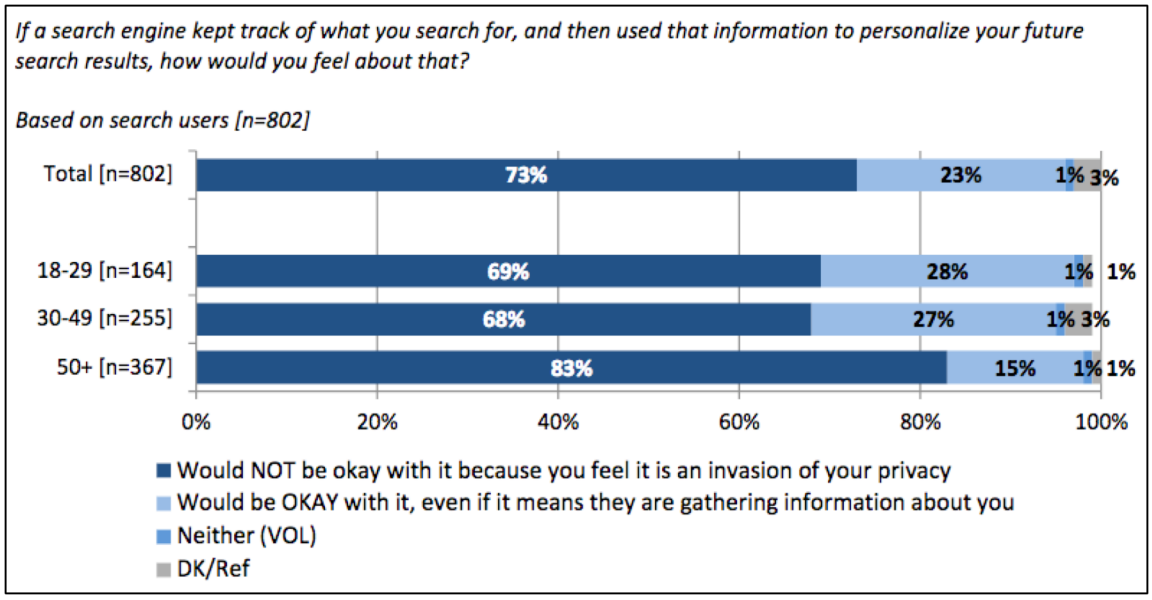

Fig. 2. Three-quarters of search users say collecting user information to personalize search results is not okay. Source: (Kristen Purcell, 2012) Survey, January 20-February 19,2012. N=2,253 adults, age 18 and older, including 901 cell phone interviews. Interviews conducted in English and Spanish.

[1] In online marketing terminology, conversion usually means the act of converting Web site visitors into paying customers.

It comes as no surprise then that 30 percent of the users of these services are concerned about the fact that their behaviour is scrutinized without their knowledge or consent (Kristen Purcell, 2012). The increasing concerns about Web-search privacy is reflected by multiple studies. From 2014 to 2015, the interest in privacy-related issues increased a five percent (Penn, 2015), showing the negative perception that users have about personalization technologies. Lastly, a survey by the Mozilla Foundation indicates that almost a third of users feel that they have no control over their personal information on the Internet (MozillaFoundation, 2017).

We believe that the solution to those problems necessarily implies giving users real control over their data, and that this can only be achieved through technologies that strike a good balance privacy and personalization. However, when the recipient of sensitive information (i.e., the search engine) is not fully trusted, privacy protection faces a dilemma of great practical relevance.

1.1. Contribution

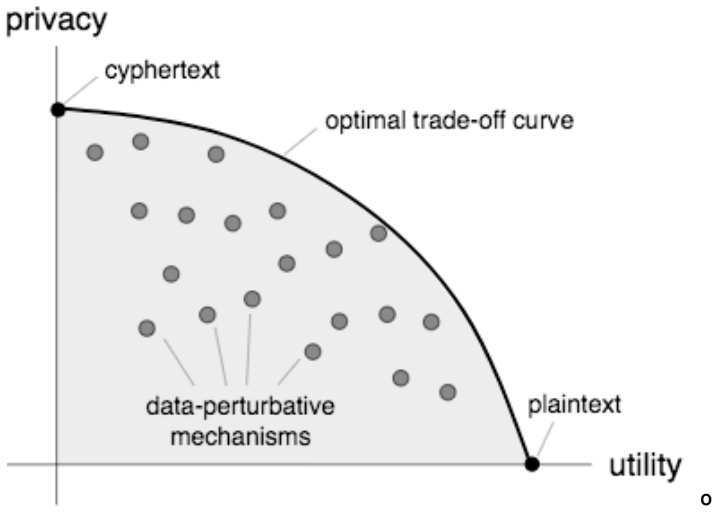

Fig. 3. Trade-off between Privacy and search-accuracy. Source: (Parra-Arnau, 2013)

The aim of this work is to contribute to the development of privacy-enhancing technologies (PETs) that may attain a suitable trade-off between privacy and

search accuracy. In particular, this thesis investigates a privacy mechanism that capitalizes on the *generalization of queries*, that is to say, the replacement of specific and probably sensitive queries, into more general, albeit semantically similar, search terms.

In the literature, a couple of previous works tackle the problem of query generalization in Web search (David Sánchez, 2013), (A. Avi, 2013). The major disadvantage of these few existing approaches, however, is that **(i)** they do not aim to protect individual queries and may reveal the actual search terms; and, more importantly, **(ii)** they are not intended for end-users, i.e., they are not designed to be used as stand-alone systems, without the need for infrastructure, and in real time. To the best of our knowledge, our work is the first to design and implement the principle of query generalization as a tool for end-users.

The designed tool, called *PrivacySearch*, is implemented as a Web-browser extension and allows users to generalize their search queries in an automated fashion, as they type the query, without consulting any external entity or database, and according to simple and intuitive privacy criteria. Our generalization algorithm is specifically contrived to satisfy various requirements in terms of computational overhead and storage, which enable the operation of the whole system in real-time and on the user side. The ultimate goal of our tool is to provide users with certain guarantees in terms of privacy and search experience.

1.2. Plan of this thesis

The reminder of this thesis is organized as follows. Sec. 2 reviews the state of the art relevant to this work. Sec. 3 describes the design principles, the system architecture and the implementation details of the proposed privacy technology. Sec. 4 evaluates different aspects of the proposed tool and shows its technical feasibility. Finally, conclusions are drawn in Sec. 5.

2. State of the Art

Numerous approaches have been proposed to protect user privacy in the context of Web search. These approaches fundamentally suggest collaboration strategies among a group of peers, and the perturbation of user data.

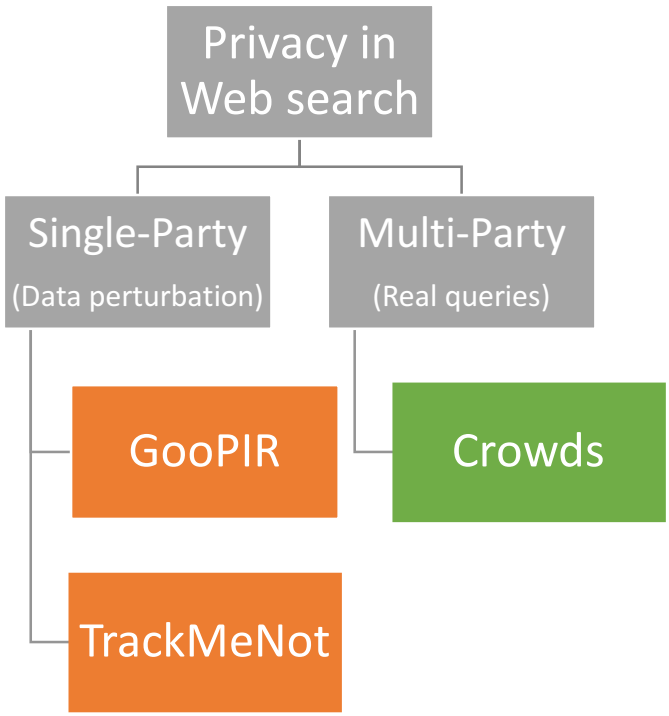

Fig. 4. Schema representing privacy techniques in Web search. Orange boxes represent existing applications; green boxes represent proposals.

An archetypical example of user collaboration is the Crowds protocol (M. Reiter, 1998). This protocol is particularly helpful to minimize requirements for infrastructure and trusted intermediaries such as pseudonymizers, or to simply provide an additional layer of anonymity. In the Crowds protocol, a group of users collaborate to submit their messages to a WSE, from whose standpoint they wish to remain completely anonymous. In simple terms, the protocol works as follows. When sending a message, a user flips a biased coin to decide whether to submit it directly to the recipient, or to send it to another user, who will then repeat the randomized decision.

Crowds provides anonymity from the perspective of not only the final recipient, but also the intermediate nodes. Therefore, trust assumptions are essentially limited to fulfilment of the protocol. The original proposal suggests adding an initial forwarding step, which substantially increases the uncertainty of the first sender from the point of view of the final receiver, at the cost of an additional hop. As in most ACSs, Crowds enhances user anonymity but at the expense of traffic overhead and delay.

An alternative to hinder an attacker in its efforts to precisely profile users consists in perturbing the information they explicitly or implicitly disclose when communicating with a WSE. The submission of false data, together with the user's genuine data, is an illustrative example of data-perturbative mechanism. In this kind of mechanisms, the perturbation itself typically takes place on the user side. This means that users need not trust any external entity such as the WSE, the Internet service provider or their neighboring peers. Obviously, this does not signify that data perturbation cannot be used in combination with other trusted-third solutions or mechanisms relying on user collaboration. It is rather the opposite—depending on the trust model assumed by users, this class of technologies can be synergically combined with any of other approach. In any case, data-perturbative techniques come at the cost of system functionality and data utility, which poses a trade-off between these aspects and privacy protection.

An interesting approach to provide a distorted version of a user's profile of interests is query forgery. The underlying idea boils down to accompanying original queries or query keywords with bogus ones. By adopting this data-perturbative strategy, users prevent privacy attackers from profiling them accurately based on their queries, without having to trust neither the service provider nor the network operator, but clearly at the cost of traffic overhead. In other words, inherent to query forgery is the existence of a trade-off between privacy and additional traffic.

A software implementation of query forgery is the Web browser add-on TrackMeNot (D. C. Howe, 2006). This popular add-on makes use of several strategies for generating and submitting false queries. Basically, it exploits RSS feeds and other sources of information to extract keywords, which are then used to generate false queries. The add-on gives users the option to choose how to forward such queries. In particular, a user may send bursts of bogus queries, thus mimicking the way people search, or may submit them at predefined intervals of time. Despite the strategies users have at their disposal, TrackMeNot is vulnerable to a number of attacks that leverage on the semantics of these false queries as well as timing information, to distinguish them from the genuine queries (Richard Chow, 2009).

GooPIR (Josep Domingo-Ferrer, 2009) is another proposal aimed at obfuscating query profiles. Implemented as a software program, this approach enables users to conceal their search keywords by adding some false keywords. To illustrate how this approach works, consider a user wishing to submit the keyword “depression” to Google and willing to send it together with two false keywords. Based on this information, GooPIR would check the popularity of the original keyword and find that “iPhone” and “elections” have a similar frequency of use. Then, instead of submitting each of these three keywords at different time intervals, this approach would send them in a batch. The proposed strategy certainly thwarts attacks based on timing. However, its main limitation is that it cannot prevent an attacker from combining several of these batches, establishing correlations between keywords, and eventually inferring the user’s real interest (Ero Balsa, 2012). As an example, suppose that the user’s next query is “Prozac” and that GooPIR recommends submitting it together with the keywords “shirt” and “eclipse”. In this case, one could easily deduce that the user is interested in health-related issues.

Another form of perturbation, which is the one considered in this work, is tackled in (A. Avi, 2013), (David Sánchez, 2013). Given a query corresponding to an intended interest, (A. Avi, 2013) generates a set of more general, semantically-related queries that loosely correspond to that interest. Each of these queries are

submitted independently to the WSE, and the level of privacy protection is determined by the least private term. Upon receiving all search results, the proposed system tries to reconstruct a ranking similar to the one that the query would have yielded. However, with the increasingly sophisticated tracking technologies available these days, it is likely that the WSE can reverse the procedure and obtain the actual interest by combining all the scrambled queries. Consequently, the submission of multiple topic-related queries may improve accuracy, but it may not protect the true specific interest from the service provider.

Similarly, (David Sánchez, 2013) proposes replacing user queries with general terms. However, since the aim is not to protect each single search but the accumulated query profile, it may happen that certain individual queries are exposed to the service provider. Besides, (David Sánchez, 2013) poses evident implementation and security issues, which prevent them from being put into practice as user tools. For example, the cited work relies on an external database for generalizing queries, and applies computationally-intensive natural language processing techniques.

3. PRIVACYSEARCH - LOCAL GENERALIZATION OF WEB SEARCHES IN REAL TIME

In this section, we present the main contribution of this work: a privacy system that allows users to distort their query profile by protecting each single query from the standpoint of a malicious WSE.

Our approach is based on the principle of query generalization. Although there exist theoretical proposals relying on this same principle of information perturbation (A. Avi, 2013), (David Sánchez, 2013); there is no practical tool available to end-users, which applies this perturbation principle and effectively protects each individual search query against WSEs. As discussed in Sec. 2, the proposals available in the literature may fail in the protection of search terms and cannot be implemented as stand-alone systems. In contrast, our technology is specifically designed to meet these two fundamental requirements:

- **Real time.** At the time of sending a query, it must be replaced automatically by a term of a higher semantic category, without the user perceiving any degradation in the search engine's response time.

- **Local mode.** The generalization algorithm, and in general the query protection tool, must perform all operations on the user side, without the help of any type of infrastructure.

The few existing proposals in the literature contemplate the use of external databases to carry out query processing and/or generalization. This is the case, for example, of (David Sánchez, 2013), which uses the Open Directory Project to determine the category to which a query belongs. However, querying external databases is not an appropriate solution. An attacker, possibly the database itself, could leverage the queries to profile the user in question and compromise their privacy.

In this work, we present *PrivacySearch*, a technology that is aimed to address the issues raised in Sec. 2 and meet the requirements of real time and local-mode mentioned above. This tool is aimed at users concerned about their online privacy, who wish to prevent search engines like Google from building a precise profile based on their queries.

Our solution replaces the queries to be submitted by a user with generic terms, so that the search engine cannot find out the exact information they are looking for. How generic these terms are determined by the users themselves through appropriate and simple privacy configurations.

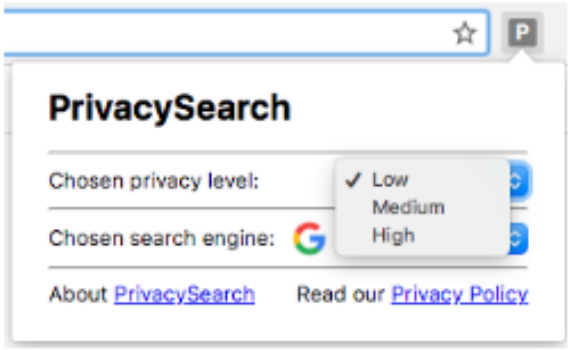

Fig. 5. Selection of the level of privacy in our tool.

Specifically, our tool allows users to configure three levels of privacy: low, medium and high (see Fig. 5). The selected privacy level indicates how generic the query generated by PrivacySearch will be from the original query[2]. As an example, consider the case in which a user wants to send the specific query “bipolar disorder” on a topic as sensitive as health. For a low level of privacy, our tool would not modify the query; with a medium level of protection, PrivacySearch would send the query “mental disorder”; and for the highest protection level, the tool would return the term “health”.

[2] Users can also choose in which search engine the processed search will be carried out.

3.1. System description

PrivacySearch has been developed as a plug-in for the Chrome browser[3] and is currently integrated in the navigation bar. Web searches are sent through the navigation bar after typing the keyword “privacy”.

It is worth emphasizing that the current implementation of our tool, available online in the Chrome Store, is based on a semantic ontology without using sophisticated natural language processing techniques or deep semantic analysis. As we shall describe later in this section, our aim is investigating the practical feasibility of query generalization and the performance of a tool implementing this principle.

As the user types their query, recommendations of previously processed queries may be shown to the user. The ability of our tool to operate in real time is of special relevance to conduct this task.

For the development of *PrivacySearch*, several techniques of Natural Language Processing (NLP) have been used. The NLP is a discipline of Artificial Intelligence that deals with the formulation and research of computational mechanisms for communication between people and machines through the use of Natural Languages.

To generalize user queries, we capitalize on WordNet as a categorizer (Miller, 1995). WordNet is a database that contains lexical relations between words in English. In particular, it has 117 798 names, 11 529 verbs, 21 479 adjectives and 4 481 adverbs. Since each WordNet entry stores its hyperonym and its hyponym, it can be construed as an ontology (Vaclav Snasel, 2005). For example, the term “dog” has the following hyperonym (note that entity is the common hyperonym to all WordNet terms):

---

[3] The plug-in is available at https://chrome.google.com/webstore/detail/ecippblhocppaciehgckhfboegciekkf.

⇒ Dog

⇒ Canid

⇒ ...

⇒ Animal

⇒ ...

⇒ Entity

When our plug-in receives a query, it processes it in three different steps. First, PrivacySearch pre-process the text; subsequently it performs a linguistic disambiguation; and finally it concludes by making a categorization according to the level of privacy selected by the user. Figure 5 shows the processing schema.

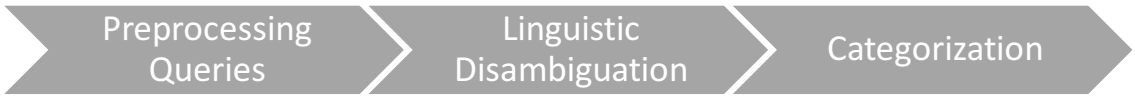

Fig. 6. PrivacySearch processing steps.

Pre-processing Queries. In this very first step, a series of tasks are conducted that aim to prepare the query for further processing by WordNet. This step is of special importance for the real-time requirement specified in the previous subsection. Essentially, queries are “simplified”, although the meaning is kept, in order to diminish the workload of the following two steps.

In fact, as we can see in Figure 7, the pre-processing step occupies more than 75% of the total execution time of the algorithm.

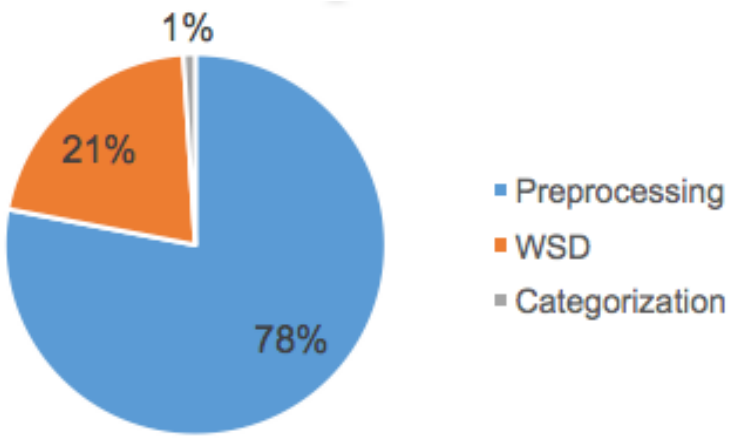

Fig. 7. Average execution time consumption for each step of the PrivacySearch processing

The pre-processing done in our tool is simpler than the one suggested in (David Sánchez, 2013). In the cited work, the authors obtain the grammatical category and perform a morpho-syntactic analysis of each term using models of maximum entropy. In our case, we perform a less complex natural language processing, without taking into account neither the grammar nor the amount of information provided by each term of a query, which significantly reduces the computational overhead and notably speeds up the response time.

To pre-process a query, we perform the following tasks:

- The query is converted to lowercase, and the score is deleted. In this way, we avoid false negatives when searching for each term in WordNet.

- The so-called “stop words” of the query are removed. The stop words are words that do not contribute any meaning to the query. This is the case of articles, pronouns and prepositions. In this step, they are eliminated to reduce the execution time, without altering the meaning of the query.

- The query is tokenized. Tokenization is the process by which the atomic units of a text are detected and isolated.

- Plural terms become singular. With this process, we aim to prevent false negatives in the categorization phase.

- The atomic units or tokens obtained are lemmatized. The lemmatization is the linguistic process by which the motto of a given term is determined. The slogan is the form that, by agreement, is accepted as representative of all the flexed forms of the same word, that is to say, the terms that the tool will find as entries in WordNet. For this step, we use the WordNet native slogan map.

- We obtain the n-grams existing in the query. An n-gram is a contiguous sequence of n elements contained in a text. The elements can be phonemes, syllables, letters, or words. For reasons of efficiency, our categorization algorithm uses unigrams and bigrams.

- Finally, terms are eliminated in languages other than English, or that do not exist in WordNet, in order to reduce the computing time in the language disambiguation step.

Linguistic Disambiguation. Once the query has been pre-processed, we must determine the correct meaning of each term. WordNet stores the different meanings that terms can have. For example, the word “bank” can refer to a pile or mass of some material, or a company dedicated to perform financial operations, among other meanings. The only way to ascertain the correct meaning is through linguistic disambiguation (Mark Stevenson, 2003).

Language disambiguation is currently an open problem, and there exist several approaches to tackle it. In our plug-in, we have implemented a linguistic disambiguation algorithm based on the Lesk algorithm (Lesk, 1986). This algorithm relies on the principle that neighbour’s words within a text tend to share a common theme. The concrete implementation of the algorithm is shown below by means of an example:

1. Let *A B C* be the input query.

2. Assume each term has a set of associated meanings, denoted as follows: A ∈ {a1, a2, a3}, B ∈ {b1}, C ∈ {c1, c2}.

3. All possible permutations of meanings are formed:

   $(a_1, b_1, c_1), (a_1, b_1, c_2), (a_2, b_1, c_1), \ldots, (a_3, b_1, c_2)$

4. A function F(a, b) is defined that returns the distance between a pair of meanings:

$$F(a,b) = \sum_{i=0}^{|H_a|} \sum_{j=0}^{|H_b|} \frac{1}{H_a/2} + \frac{1}{H_b/2}$$

   being H_a the sequence of existing hyperonyms between *a* and *entity*, and H_b the sequence of existing hyperonyms between b *and entity*, where

   $H_{ai} \neq H_{aj}$

5. F is evaluated for each pair of permutation meanings, and the permutation with less distance is chosen.

In the design of our tool, we have limited the number of possible permutations in the last step, since it increases exponentially with the number of terms and meanings of each term.

Below we illustrate with an example the linguistic disambiguation of the term bass from English. This term can refer to a musical instrument or a type of fish, among other meanings. If a user types the query “I like playing the bass guitar”, our tool should return a generic query other than if the query “I like fishing sea bass” was made. With the proposed query disambiguation algorithm, our tool is able to capture these two meanings. In particular, for the low level of privacy, the algorithm returns “musical performance guitar” for the first query, and “outdoor sport saltwater fish” for the second one.

Categorization. Once the original query has been pre-processed and the correct meanings of each *n*-gram obtained, we proceed to conduct the categorization of the resulting terms. To carry out this step, we utilize WordNet as a categorizer. WordNet is a very popular lexical database of the English language. The proposed categorization algorithm is described in Algorithm 1.

Algorithm 1: Query categorization algorithm.

Input: Q, a sequence of terms that represents a genuine preprocessed query.
Output: $Q\_Categorized$, a sequence of terms that represents a generalized query, built from the genuine query.

```
let PrivacyLevel ∈ {Low, Medium, High}, level of privacy chosen by the user.
let entity, root hypername of WordNet.
for each term t in C do
    H_t ← Sequence of direct hyperlinks from t until entity.
    if PrivacyLevel is Low then
        Q_Categorized ← H_t1
    else if PrivacyLevel is Medium then
        Q_Categorized ← H_t(|H_t|·0.1)
    else if PrivacyLevel is High then
        Q_Categorized ← H_t(|H_t|·0.2)
end
```

As we discussed at the beginning of this section, WordNet can be considered an ontology. This allows us to easily access all the hypernonyms of a term until reaching the common hyperonym to all of them, i.e., *entity*. Accordingly, for each n-gram existing in the processed query we extract a hyperonym.

As frequently done in information retrieval and text mining, our query classifier also relies on the term frequency-inverse document frequency model. Said otherwise, we weight the resulting hyperonym/s based on the frequency of occurrence of the corresponding unigrams and bigrams.

Next, we specify the rule for choosing the depth of the hyperonym/s in the hierarchy, as a function of the level of privacy:

- For the level “low”, the first hyperonym is extracted from each *n*-gram.

- For the privacy level “medium”, we compute the depth from the *n*-gram to *entity*, and the hyperonym with a depth of 10% is extracted.

- For the privacy level “high”, we compute the depth from the *n*-gram to *entity*, and the hyperonym with a depth of 20% is extracted.

A maximum depth of 20% has been configured since, for a large number of terms, experimental evidence shows that a higher percentage returns results that are too generic and, therefore, of little utility. In addition, since WordNet is not an ontology per se (it contains redundancies in its hierarchy), we proceed as follows: to avoid cycles when categorizing, when one hyperonym is detected, the following available secondary hyperonym is chosen and not the direct one.

3.2. Implementation Details

For implementing the system described in the previous section, several options were considered. One of them was OpenSearch, a standard that allows publishing search results in a format suitable for syndication and aggregation, so web pages and search engines are able to publish their results in an accessible way.

Using OpenSearch is simple, it is defined in an xml file that specify which output URL is generated for each input query, in this way, we would obtain a search service available for all possible browsers.

The counterpart is that both the OpenSearch definition and the search service logic must be uploaded on a server. With PrivacySearch we look for all the processing to be done from the client side, so OpenSearch is not suitable for our implementation.

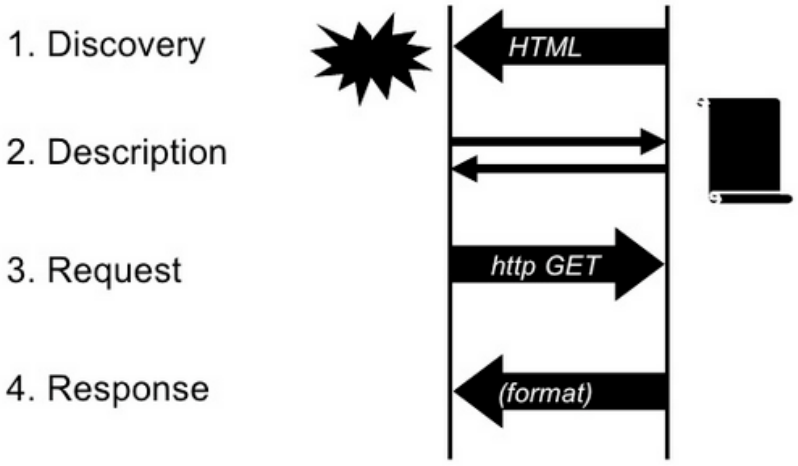

Fig 8. OpenSearch implementation schema.

Once OpenSearch has been discarded, the only option left consists on developing an extension for a specific browser. We chose Google Chrome since its API to develop extensions is simple and widely documented, and also, it is the browser with the highest market share, having more than half of the market share.

On the implementation for Google Chrome it is important to point out that we only keep two elements in the browser session memory: **(i)** the level of privacy chosen by the user (by default low), **(ii)** the search engine chosen by the user (by Google). No genuine search is sent to any server, nor does it persist in the user's browser or computer.

The WordNet database is stored in the extension per se. It should be mentioned that for efficiency in both execution time and memory, the original database is not used.

Two important changes were made to WordNet. First, non-relevant information such as definitions, hyponyms, multiple flags, etc. was eliminated in order to minimize the size of the database. Once minified we converted it into Javascript maps so that retrieving elements from it was as fast as possible.

In appendix A we explain in more detail the structure and logic of our implementation.

4. EVALUATION

In this section, we evaluate our tool in terms of computational efficiency and performance, with the aim of verifying whether the design requirements specified in Sec. 3 are met. Due to the relevance of showing recommended queries (i.e., generalized terms) in real time, we have performed an analysis of the execution times obtained with selected real queries from a database.

The database of queries we have employed in our experiments was published by the AOL search engine in 2006 (Arrington, 2006). This database contains about 37 million queries of 657 000 unique users, obtained during a period of three months (from 1 March 2006 to May 31, 2006). Other databases are available such as those of Altavista (Beales, 2010) and MSN database (Zhicheng Dou, 2007), but since they are rather similar in terms of user queries, we restrict just to the AOL data set.

To carry out our analysis, we run PrivacySearch on the first 100 000 records of the database. The selected subset includes searches made between March 1 to March 6, 2006. Tables 1 and 2 respectively show some statistics of the data set under study, and those of the data after the pre-processing phase.

1 term in the query:	30 603	**Max.**	25
2 terms in the query:	26 149	**Mean**	2.65
3 terms in the query:	18 750	**Standard deviation**	1.79
4 terms in the query:	11 195		
5 terms in the query:	6 415		
>5 terms in the query:	6 888		

Table 1. Some statistics of the first 100 000 queries of the AOL query database.

0 terms in the query:	29 502	**Max.**	12
1 term in the query:	21 731	**Mean**	1.56
2 terms in the query:	24 859	**Standard deviation**	1.41
3 terms in the query:	14 939		
4 terms in the query:	6 082		
5 terms in the query:	1 925		
>5 terms in the query:	962		

Table 2. Some statistics of the first 100 000 queries, after pre-processing.

One of the effects of the pre-processing step is an average reduction of 41.13% in the number of terms of per query. This significant reduction is essentially due to the elimination of stop words, the deletion of terms without an entry in WordNet, and the grouping of terms in n-grams. Furthermore, around 30% of queries have been omitted, since a number of them either referred to URLs, or included character names and/or terms, or were written in a language other than English; the current version of our tool only works for English searches.

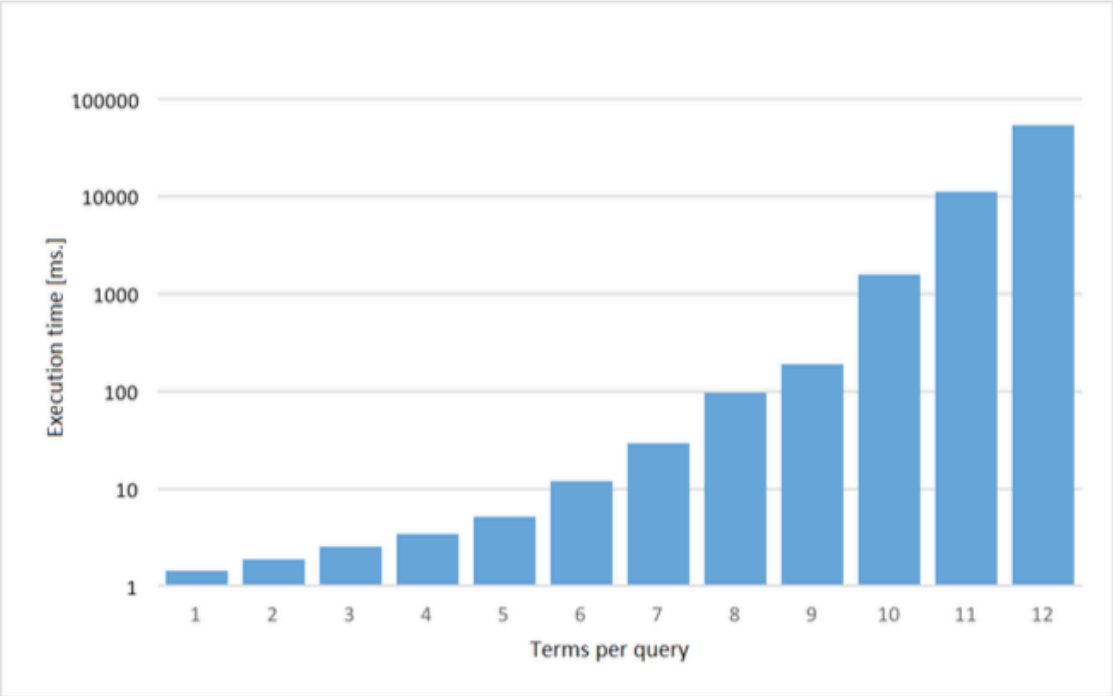

Fig. 9. Average execution time based on the number of terms.

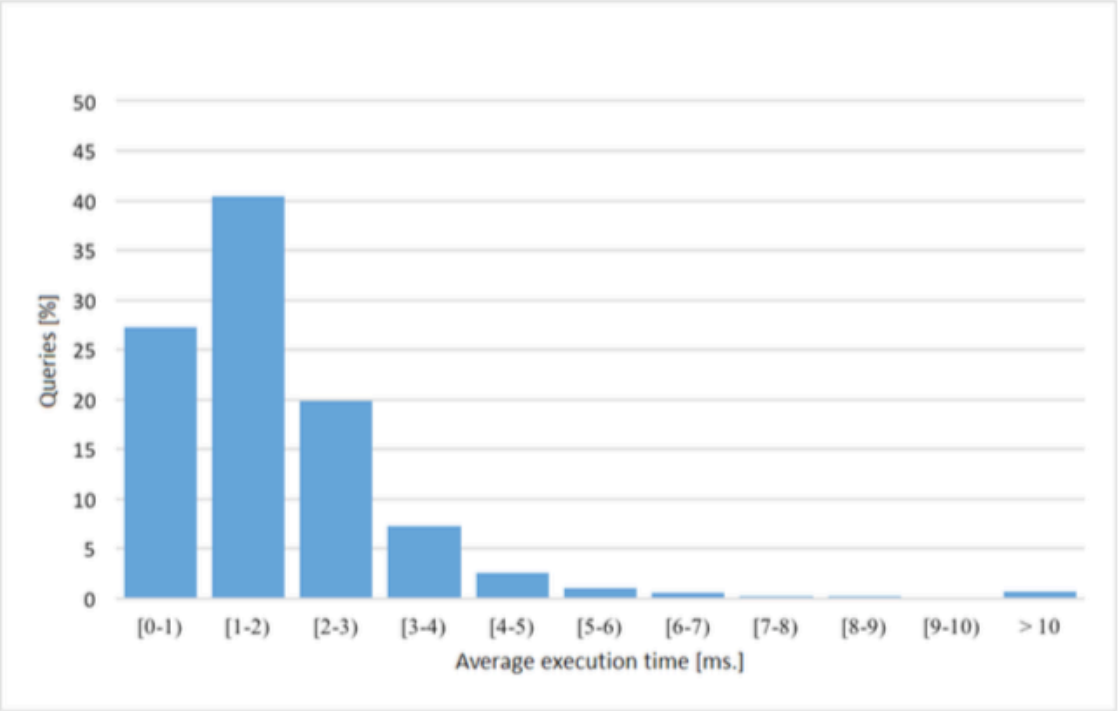

Fig. 10. Average execution time.

Our experimental results are shown in Figs. 9 and 10. In the former figure, we observe that, in more than 85% of cases, the tasks of pre-processing, disambiguation and categorization were performed in less than 3ms. This result is of special relevance as it demonstrates that privacy protection may come at the cost of negligible processing overhead. In the latter figure, on the other hand, we can appreciate that the average running time increases exponentially with the number of terms. The reason is due to the fact that the number of computations performed by the linguistic disambiguation algorithm depends on the number of terms in the query and the number of meanings per search term.

Despite this, if we consider only the pre-processing phase, the average number of terms per query in the selected subset of data yields 2.65; in the Altavista and BIEW databases, this number becomes 2.35 and 1.63, respectively. This, together with the fact that 99.93% of the executions carried out in our analysis did not exceed 10ms, allows us to conclude that our tool performs suitably for real-time use. Last but not least, as far as memory use is concerned, our extension occupies 4.3MB approximately once packaged, and 51.6MB once installed in Chrome and in use.

5. Conclusions and Future Work

The use of personalization techniques by WSEs is a promising way to improve the quality of searches. However, these techniques lend themselves to the construction of profiles of interests and preferences, which pose serious concerns to user privacy. This work focuses on a data-perturbative mechanism by which specific queries are transformed into more general terms (although semantically similar) and so less sensitive. Although there exist few proposals based on query generalization, no solution has been designed nor developed that brings this principle into practice and is intended for end-users.

In this thesis, we have proposed *PrivacySearch*, a browser tool that allows users to generalize the queries sent to a search engine, automatically, without the need of any type of infrastructure or external database, and in real time. With PrivacySearch, users can control the specificity of their interest profiles in front of a search engine, through a flexible and intuitive control of the sensitivity of the information they are disclosing. In contrast to other approaches, our tool protects each individual query independently and, as such, does not make any assumption on the ability of the WSE to track all them. Experimental results show that our tool is able to categorize complex searches in real time while users types their queries, without affecting the performance of the system.

Future work will evaluate the proposed tool further with real users, and attempt to determine the utility loss incurred by the generalized queries (e.g., how many result pages a user must go through to find the link that best fits the original query).

APPENDIX A: PRIVACYSEARCH IMPLEMENTATION STRUCTURE

The structure of the implementation of PrivacySearch is the following one:

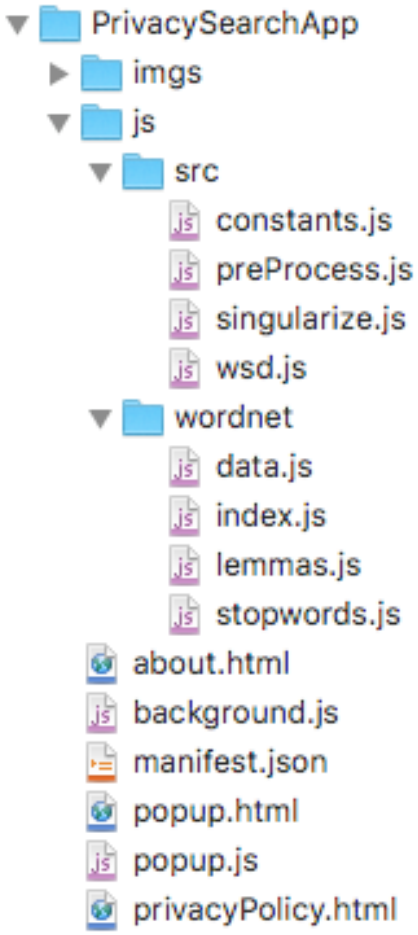

Fig 11. PrivacySearch implementation structure.

In the directory js/src the files in Javascript that contain the logic of our extension are stored:

- **constants.js** - Stores all the constants used, such as the depth of searches with medium and high privacy level, the index to the root node of WordNet, etc.

- **preProcess.js** - It contains all the logic for pre-processing queries.

- **singularize.js** - Map and code used for singularize terms.

- **wsd.js** - This file has the word sense disambiguation algorithm implementation, and also the logic used in the categorization of queries.

In the js/wordnet directory, the simplified database based on WordNet is stored:

- **data.js** - This file stores all WordNet terms together with its hyperonyms, without distinguishing between names, verbs, adjectives, etc.

- **index.js** - It contains (term - index) tuples.

- **lemmas.js** - Map with English lemmas, used in the lemmatization process.

- **stopwords.js** - List of English stop words.

In the root directory we have included all the files that are related to the extension's deployment and visualization.

- **about.html** - The ‘About PrivacySearch’ view.

- **privacyPolicy.html** - The ‘PrivacyPolicy’ view.

- **popup.html & popup.js** - Main view that is shown to the user, it also controls the changes of parameters made by the user.

- **background.js** - This file controls the searches, recovering from session the level of privacy, and the selected search engine, to carry out the desired search by the user.

- **manifest.json** - It is information that Google Chrome needs from the extension (where the logic files are, what permissions are needed, etc.).

FIGURES INDEX

TABLES INDEX

REFERENCES

- A. Avi, P. S. (2013). A query scrambler for search privacy on the internet. *Information Retrieval , 16* (6), 657-679.
- Adan Ortiz-Cordova, a. B. (2012). Classifying web search queries to identify high revenue generating customers. *Journal of the Association for Information Science and Technology , 63* (7), 426-1441.
- Adar, E. (2007). User 4xxxxx9: Anonymizing query logs. *Proc of Query Log Analysis Workshop, International Conference on World Wide Web* .
- Arrington, M. (2006). AOL Proudly Releases Massive Amounts of Private Data. *TechCrunch* .
- Beales, H. (2010). The value of behavioral targeting,. *Tech. rep., Netw. Advertising Initiative* .
- Bin Cao, J.-T. S. (2017). PQC: Personalized Query Classification. *ACM Eighteenth Conference on Information and Knowledge Management* , 1217-1226.
- Craig Silverstein, M. H. (1999). Analysis of a very large web search engine query log. *ACm SIGIR Forum , 33* (1), 6-12.
- D. C. Howe, H. N. (2006). TrackMeNot: Resisting surveillance in web search. *Lessons from the Identity Trail: Privacy, Anonymity and Identity in a Networked Society , 290*, 417-436.
- David Sánchez, J. C.-R. (2013). Knowledge-Based Scheme to Create Privacy-Preserving but Semantically-Related Queries for Web Search Engines. *Information Sciences , 218*, 17-30.
- Elif Aktolga, A. J. (2013). Building Rich User Search Queries Profiles. *International Conference on User Modeling, Adaptation, and Personalization* , 254-266.
- Ero Balsa, C. T. (2012). OB-PWS: Obfuscation-based private web search. *Security and Privacy (SP), 2012 IEEE Symposium* , 491-505.
- European Commission. (2016). Media pluralism and democracy: outcomes of the 2016 Annual Colloquium on Fundamental Rights. *2016 Annual Colloquium on Fundamental Rights* , 14-15.

- Josep Domingo-Ferrer, A. S.-R. (2009). h(k)-Private information retrieval from privacy-uncooperative queryable databases. *Online Information Review , 33* (4), 720-744.
- Kristen Purcell, J. B. (2012). Search Engine Use 2012. *Pew Internet* .
- Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation* , 24-26.
- M. Reiter, A. R. (1998). Crowds:anonymityforWebtransactions. *ACMTrans.Inf. Syst. Secur. , 1*, 66-92.
- Matwins, S. S. (1998). Text classification using WordNet hypernyms. *Usage of WordNet in Natural Language Processing Systems* .
- Miller, G. (1995). WordNet: A lexical database for English. *Communications of the ACM , 38* (11), 39-41.
- Mozilla_Foundation. (2017). Online Privacy & Security Survey.
- Pariser, E. (2017). El filtro burbuja: Cómo la web decide lo que leemos and lo que pensamos.
- Penn, M. (2015). Views from Around the Globe: 2nd Annual Poll on How Personal Technology is Changing Our Lives.
- Prasanna Ganesan, H. G.-M. (2003). Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems (TOIS) , 21* (1), 64-93.
- Richard Chow, a. P. (2009). Faking contextual data for fun, profit, and privacy. *Proceedings of the 8th ACM workshop on Privacy in the electronic society* , 105-109.
- Trevor Mansuy, a. R. (2006). A characterization of WordNet features in Boolean models for text classification. *Proceedings of the fifth Australasian conference on Data mining and analytics , 61*.
- Vaclav Snasel, P. M. (2005). WordNet ontology based model for web retrieval. *Web Information Retrieval and Integration* , 220-225.
- Wilks, M. S. (2003). Word-SenseDisambiguation. *TheOxfordHandbook of Computational Linguistics* , 249-265.

- Xuehua Shen, B. T. Privacy protection in personalized search. *ACM SIGIR Forum , 41* (1), 2007.
- Yabo Xu, K. W. (2007). Privacy-enhancing personalized web search. *Proceedings of the 16th international conference on World Wide Web* , 591-600.
- Zetter, K. (2009). Yahoo Issues Takedown Notice for Spying Price List. *Wired* .
- Zhicheng Dou, R. S.-R. (2007). A large-scale evaluation and analysis of personalized search strategies. *Proceedings of the 16th international conference on World Wide Web* , 581-590.