Inf Retrieval (2013) 16:657–679
DOI 10.1007/s10791-012-9212-1

A query scrambler for search privacy on the internet

Avi Arampatzis · Pavlos S. Efraimidis · George Drosatos

Received: 28 September 2011 / Accepted: 24 September 2012 / Published online: 9 October 2012

Abstract We propose a method for search privacy on the Internet, focusing on enhancing plausible deniability against search engine query-logs. The method approximates the target search results, without submitting the intended query and avoiding other exposing queries, by employing sets of queries representing more general concepts. We model the problem theoretically, and investigate the practical feasibility and effectiveness of the proposed solution with a set of real queries with privacy issues on a large web collection. The findings may have implications for other IR research areas, such as query expansion and fusion in meta-search. Finally, we discuss ideas for privacy, such as k-anonymity, and how these may be applied to search tasks.

Keywords Query scrambler · Search privacy · WordNet · Fusion

1 Introduction

The Internet has gradually become the primary source of information for many people. More often than not, users submit queries to search engines in order to locate content. Considering the Internet as a huge library, web-search corresponds to a search within this library. While conventional library records are private under law, at least in the U.S., Internet users might be exposed by their searches.

Every time a user submits a query to a web search engine, some private information about the user and her interests might be leaked with the query. The query representing the interest will be saved in the engine's session-logs, or it may be intercepted by the Internet

A. Arampatzis (✉) · P. S. Efraimidis · G. Drosatos
Department of Electrical and Computer Engineering, Democritus University of Thrace,
University Campus, 67100 Xanthi, Greece
e-mail: avi@ee.duth.gr

P. S. Efraimidis
e-mail: pefraimi@ee.duth.gr

G. Drosatos
e-mail: gdrosato@ee.duth.gr

Table 1 Queries which may have privacy issues

welfare fraud	post traumatic stress
rehabs in harrisburg pa	herpes
how to make bombs	lawyers for victims of child rape
hazardous materials	acute hepatitis
gun racks	police scanner

provider or any other node in the network path. Table 1 presents some queries, which—depending on culture, country laws, or corporation rules—may have privacy issues. Some of those queries may correspond to malicious intentions, but we will not distinguish.

There is ongoing research on web-log anonymization, which has turned out to be a non-trivial problem. The use of fairly advanced techniques like token-based hashing (Kumar et al. 2007) and query-log bundling (Jones et al. 2008) shows that web-log anonymization is by far not solved. Another server-based approach for anonymizing query logs is based on micro-aggregation Erola et al. (2011). The above approaches require the user to trust the good intentions of the search engine (with respect to the user's privacy) and additionally to tolerate the inevitable possibility of personal data leakage of the server-based methods. Thus, it currently makes sense to investigate the issue also from the other side: *how users can protect themselves*.

In September 2006, AOL released a collection with search query-log data containing about 21 million web queries collected from about 650 thousand users over 3 months (Pass et al. 2006). To protect user privacy, each real IP address had been replaced with a random ID number. Soon after the release, the first 'anonymous' user had been identified from the log data. In particular, the user given the ID 4417749 in AOL's query-log was identified as the 62-old Thelma (Barbaro and Zeller 2006 (accessed June 3, 2010)). Interestingly, this identification was based solely on the queries attributed to her ID. Even though AOL withdrew the data a few days after the privacy breach, copies of the collection still circulate freely online. The incident only substantiated what was already known: web search can pose serious threats on the privacy of Internet users.

There are some countermeasures a common user can take to protect her privacy. One is to submit the query anonymously by employing standard tools, like the Tor network[1] or some anonymization proxy. This might seem as a step in right direction, but it does not solve the privacy problem. In the AOL incident, the origin of each query was hidden, since each IP address was replaced with a random ID. However, all queries originating from the same IP were assigned the same ID. This linkability between queries submitted by the same user, resolutely increased the leakage of personal data from her query set and led to the exposition of Thelma and possibly other users. Consequently, a further step would be to make the queries of a user unlinkable. To accomplish this, a user has to continuously change her IP address and to cancel out several other information leak issues that may originate elsewhere, e.g., cookies and embedded javascript.

Alternatively or in parallel, a user can try to obfuscate her 'profile' by submitting some additional random queries. In this way, the real queries are hidden in a larger set, and the task of identifying the actual interests of the user is hindered to some extent. The TrackMeNot add-on (Howe and Nissenbaum 2009) for the Firefox browser implements such a feature. Another interesting add-on is OptimizeGoogle which, among other features, trims information leaking data from the interaction of a user with Google. An interesting

[1] http://www.torproject.org

combination of anonymization tools is employed in the Private Web Search tool (Saint-Jean et al. 2007), which is also available as an (outdated[2]) Firefox add-on. An interesting recent Firefox Add-on is Google Privacy, which removes the redirected links from the web search results. While this does not protect the user query, it helps to prevent the monitoring of which of the search results the user will actually retrieve. A community-based approach to support user privacy in information retrieval is presented in Domingo-Ferrer et al. (2009a); a user gets her query submitted by other users of a peer-to-peer community.

An interesting approach for improving search privacy was presented in Domingo-Ferrer et al. (2009b), where a single-term query of a user is mixed with a set of $k - 1$ random search terms. This approach achieves at most *k-anonymity*, which means that each keyword can be assumed to be the actual keyword with probability of $1/k$. In our view, the concept of k-anonymity provides a handy tool to quantify privacy. However, as it is applied in Domingo-Ferrer et al. (2009b) it raises practical issues; the number of terms in a search query is often bounded; for example, Google's API allows a maximum of 32 keywords. The problem further escalates for multi-term queries, where the mixed query consists of k multi-term expressions. Another related work is the plausibly deniable search technique of Murugesan and Clifton (2009) where a query is transformed into a canonical form and then submitted along with $k - 1$ appropriately selected cover queries. A survey on issues and techniques for preserving privacy in web-search personalization is given in Shen et al. (2007).

There is an important reason why the above tools and methods alone might be inadequate: in all cases, the query is revealed in its clear form. Thus, privacy-enhancing approaches employing proxies, anonymous connections, or k-anonymity, would not hide the *existence of the interest* at the search engine's end or from any sites in the network path. In addition, using anonymization tools or encryption, the plausible deniability against the *existence of a private search task* at the user's end is weakened. *Plausible deniability* is a legal concept which refers to the lack of evidence proving an allegation.[3] If a query is never disclosed to the network (never leaves the user's device/computer), then the user can deny the information need it represents. Such a denial may be deemed credible, believable, or else, plausible, due to the lack of sufficient evidence of the contrary. One way to achieve plausible deniability is to submit other related—but less exposing—queries instead, such that each of the latter queries is pointing to many plausible information needs. A related application of the notion of plausible deniability can be found in the aforementioned work of Murugesan and Clifton (2009).

Finally, there is also the related field of Private Information Retrieval (PIR). In PIR, the main problem addressed is to retrieve data from a database without revealing the query but only some encrypted or obfuscated form of it, e.g., see Yekhanin (2010); Ostrovsky and Skeith (2007); Chor et al. (1997). An interesting approach for private information retrieval that combines homomorphic encryption with the embellishment of user queries with decoy terms is presented in Pang et al. (2010). Another work in this line of research is the secure anonymous database search system presented in Raykova et al. (2009). However, all the above PIR methods have an important limitation: they assume collaborative engines.

In view of the limitations of the aforementioned approaches, we define the Query Scrambling Problem (QSP) for privacy-preserving web search as: Given a query for a web search, it is requested to obtain related web documents. To achieve this, it is allowed to

[2] The Private Web Search (PWS) tool is a Firefox Add-on. It is available on-line but seems not to be further developed. Its latest version is v0.4.2, which supports Firefox up to version 2. The PWS as well as the TrackMeNot tool have been developed in the context of the Portia project (http://crypto.stanford.edu/portia/).

[3] http://en.wikipedia.org/wiki/Plausible_deniability

interact with search engines, but without revealing the query; the query and the actual interest of the user must be protected. The engines cannot be assumed to be collaborative with respect to user privacy. Moreover, the amount of information disclosed in the process about the query should be kept as low as possible.

To address QSP, we propose the QueryScrambler; in a nutshell, it works as follows. Given a query corresponding to the intended interest, we generate a set of *scrambled queries* corresponding loosely to the interest, thus blurring the true intentions of the searcher. The set of scrambled queries is then submitted to an engine in order to obtain a set of top-n result-lists which we call *scrambled rankings*. Given the scrambled rankings, we attempt to reconstruct, at the searcher's end, a ranking similar to the one that the query would have produced, which we call *target ranking*. The process of reconstruction we call *descrambling*. Figure 1 depicts the architecture of such a system.

The novelty of the QueryScrambler is that it does not reveal the important terms of the exposing query, but it employs semantically related and less exposing terms. The amount of privacy gained can be controlled by users via a parameter which determines the minimum semantic distance between the intended query and each of the scrambled queries issued. In this respect, the QueryScrambler only protects the query against query-logs or sites in the network path. Thus, an adversary with knowledge of the method and access to all (or many) of the scrambled queries of a particular scrambled search could potentially reverse the procedure getting to the actual interest, nevertheless, this is easy to fix. In practice, the QueryScrambler can—and should—be combined with other orthogonal methods, such as those mentioned earlier. Especially, adding random queries and/or querying via multiple proxies/agents can make adversarial descrambling nearly impossible.

Inevitably, the QueryScrambler introduces an overhead over traditional web-search. We are currently not interested in its efficiency, as long as its requirements are within the reaches of current commodity desktop systems and retail Internet speeds. What we are interested in is its feasibility, focusing on the trade-off between privacy and quality of retrieved results: the method may be lossy, in the sense that the quality of results may degrade with enhanced privacy.

The rest of this paper is organized as follows. In Sect. 2, we introduce a model for generating a set of scrambled queries given a query, and investigate methods for descrambling rankings. In Sect. 3, we evaluate the effectiveness of the scrambler for different parameter values and descrambling methods. In Sect. 4, we provide a further analysis of the results, and based on observations, identify parts of the approach which may have a negative impact on retrieval effectiveness and suggest future improvements. In Sect. 5, we discuss some more concepts related to private search, and investigate whether we

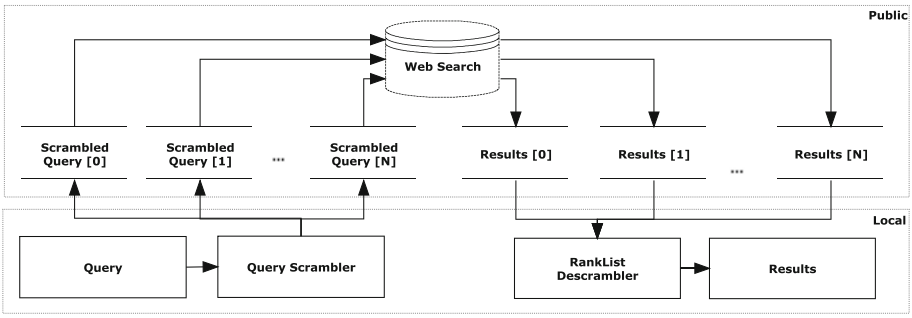

Fig. 1 Architecture of a privacy-enhancing search system

achieve our privacy goals. Conclusions and directions for further research are given in Sect. 6.

2 A query scrambler

The proposed QueryScrambler is based on a semantic framework (Sect. 2.2). First we discuss feasibility issues.

2.1 Theoretical and practical feasibility

There is no question of the theoretical feasibility of a near lossless QueryScrambler. Suppose we submit to the engine scrambled queries consisting of very frequent words, e.g., *near* stop-words. A few such scrambled queries could cover almost all the collection, which then could be downloaded to the user's site, indexed, and searched with the query. Accounting for the difference between the retrieval models, that of the engine's (usually proprietary) and that of the user's, a near-target or satisfactory ranking could be produced locally without revealing the user's target interest. In reality, such a procedure would be highly impractical or impossible for large web search engines.

Having established the theoretical feasibility of near lossless solution to QSP with the procedure described above, what we are interested in is the trade-off between the *descrambled ranking quality* and the following three quantities:

1. *scrambling intensity*, i.e., the minimum semantic distance between the query and the set of scrambled queries,
2. *query volume*, in terms of the cardinality of the scrambled query set, and
3. *ranking depth*, i.e., the number of results returned by the engine for a scrambled query.

The scrambling intensity represents the degree of hiding the true intentions; it should be given the highest priority and be kept high, affecting negatively the ranking quality. Query volume and ranking depth have the largest impact on the practical feasibility of the task; they should be kept low, affecting again negatively the ranking quality.

In practice, web search engines usually do not return the full set of results, but truncate at some rank n. For example, the Google API returns a maximum of top-1000 results per query. In this respect, we could eliminate the depth from the parameters by setting it to top-1000, a rather sufficient and practical value.

2.2 A semantic framework

Simplifying the analysis, let us assume that a query represents a single concept. Concepts more general to the query, i.e., *hyper-concepts*, would completely cover the query's concept, as well as other concepts. In this respect, some other query representing one of the hyper-concepts of the query would target more results than the query but include all results targeted by the query. Privacy for the query can be enhanced by searching for any of the hyper-concepts instead and then filtering the results for the query concept. Thus, queries representing hyper-concepts of the query can be used as scrambled queries (SQ).

Figure 2a depicts an idealized concept space. As an example consider a query Q representing the concept 'herpes' (the disease), but searching for the concept of 'infectious disease'. SQ1 could represent 'infectious disease'. SQ2 could represent 'health problem', a more general concept than this of SQ1 denoted by covering a larger area in the space. We

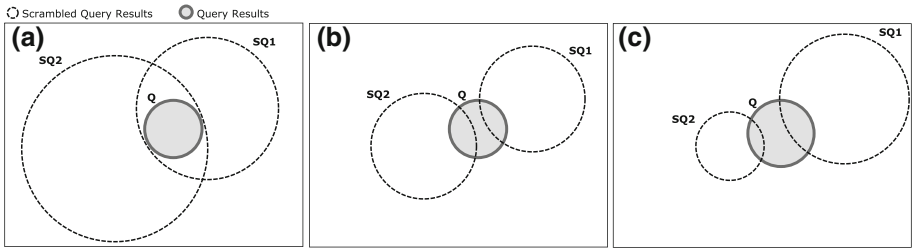

Fig. 2 Results for two scrambled queries in relation to a query Q: **a** all results in a concept space of uniform density, **b** top-n results in a uniform document space, **c** top-n results in a non-uniform document space. Q represents all relevant results

assume that the space has a uniform concept density. Both SQ1 and SQ2 cover Q completely.

Trying to transform Fig. 2a to a document space, some important issues come at play:

1. Concept retrieval via the bag-of-words paradigm is inherently noisy. Semantic relations between keywords or phrases are seldom used. Thus, using concept names as keywords, e.g., using 'infectious disease' directly as SQ1, would count on 100 % co-occurrence of this phrase on all documents containing the word 'herpes' in order to fulfil Fig. 2a.
2. Web search engines usually do not return the full set of results but truncate at some rank n.
3. Document spaces are non-uniform, a direct result of the collection at hand not covering all concepts equally.

Let us first consider an idealized uniform document space. The first issue would result to SQ1 and SQ2 circles not covering Q completely, with their centers positioned at slightly different areas (assuming keyword retrieval approximates well concept retrieval). The second issue would enforce equal circle areas for SQ1 and SQ2, denoting n results (assuming that both scrambled queries have $\geq n$ results). These are depicted in Fig. 2b.

Factoring in non-uniformity of the document space, i.e., the third issue, the picture changes to Fig. 2c; the SQ2 area is denser than the area of SQ1, denoted by the reduced area covered by n results. The size of the Q area may also change, depending on the number of relevant results in the collection. Obviously, a single SQ would not cover all results corresponding to the query, so for a full coverage multiple SQs would have to be used.

Next, we investigate theoretically the trade-off between the number of SQs used and the expected coverage. Then, we describe the current implementation of the QueryScrambler.

2.3 Scrambled query volume

One important parameter of the QueryScrambler is the number of scrambled queries that should be executed. Naturally, a larger number of scrambled queries will increase the recall. We provide a simple probabilistic argument for how the number of scrambled queries trades off with recall.

Assume that we are interested in $\ell \leq n$ target items, where n is the search engine's truncation rank. Also, assume that we manage to generate a set of scrambled queries, such that each scrambled query catches r of the target items. If for example $r = 5$, $\ell = 50$ and

$n = 1000$, then each scrambled query will retrieve (on average) $r = 5$ target items in 1000 retrieved items. The precision of the scrambled ranking will only be 0.5 %, a value which should be considered sufficiently low for protecting the user's privacy.

How many scrambled queries should we submit in order to catch, with a high probability, all target items? If we assume that the target items in the results of each scrambled query are independent random items of the set of ℓ target items (of course, in reality the items will not be independent, but we will make this simplifying assumption here to obtain an indication about the number of scrambled queries that are needed), then this problem can be modelled as a *balls and bins* problem; each target item corresponds to a bin and we throw balls randomly into the bins until all bins have at least one ball. In particular, this specific problem corresponds to *the coupon collector's problem* in which there are n types of coupons and independent random coupons are chosen until a coupon of each type has been found. The following result which gives the average number of coupons that have to be drawn in order to find all ℓ coupons is well-known; see for example Motwani and Raghavan (1995); Mitzenmacher and Upfal (2005). For completeness, we provide a short proof of it in the context of QSP.

Lemma 1. *The average number of random target items that have to be drawn in order to find all ℓ target items is $\ell\, H_\ell$, where H_ℓ is the harmonic number. The harmonic number satisfies $\ln \ell \leq H_\ell \leq \ln \ell + 1$, which implies that $H_\ell = \ln \ell + \Theta(1)$.*

Proof For $0 \leq i \leq \ell - 1$, assume that i distinct target items have been found. Let X_i be the number of random target items that have to be drawn until the next distinct target item is found. Then, the sum $Y_\ell = \sum_{i=0}^{\ell-1} X_i$ is a random variable that corresponds to the total number of random target items that are drawn until all ℓ distinct target items are found. Each random variable X_i is geometrically distributed with parameter $p_i = \frac{\ell - i}{\ell}$. Thus, the expected value of X_i is $E[X_i] = 1/p_i$ and the expected value of the sum Y_ℓ is

$$E[Y_\ell] = E[X_1 + \cdots + X_\ell] = E[X_1] + \cdots + E[X_\ell] = \ell \sum_{i=1}^{\ell} \frac{1}{i} = \ell H_\ell. \tag{1}$$

For the harmonic number H_ℓ, it holds

$$H_\ell = \sum_{i=1}^{\ell} \frac{1}{i} \leq \sum_{i=0}^{\lfloor \log \ell \rfloor} \sum_{j=0}^{2^i - 1} \frac{1}{2^i + j} \leq \sum_{i=0}^{\lfloor \log \ell \rfloor} \sum_{j=0}^{2^i - 1} \frac{1}{2^i} \leq \sum_{i=0}^{\lfloor \log \ell \rfloor} 1 \leq \log \ell + 1. \tag{2}$$

□

We now apply the above arguments to the QueryScrambler. Let Y_ℓ be the number of random target items that have to be retrieved to obtain the ℓ target items. As noted earlier, Y_ℓ is a random variable and Fig. 3 shows how its expected value $E[Y_\ell]$ is related to ℓ. For $\ell = 50$, the expected number of random items that have to be drawn to retrieve all target items is $s = 50\, H_{50} \simeq 225$. If every scrambled ranking includes $r = 5$ target items, then this implies that on average $v = 45$ scrambled queries have to be executed. This number is reduced to $v = 15$, if the scrambled queries return on average $r = 15$ target items.

To account for deviations, we may set a more conservative goal where we focus on obtaining not necessarily all distinct target items but only a (large) fraction of them. Let Z be the number of distinct random items after m random target items have been retrieved and let μ be its expected value $\mu = E[Z]$. Then, it is not hard to show the following Lemma.

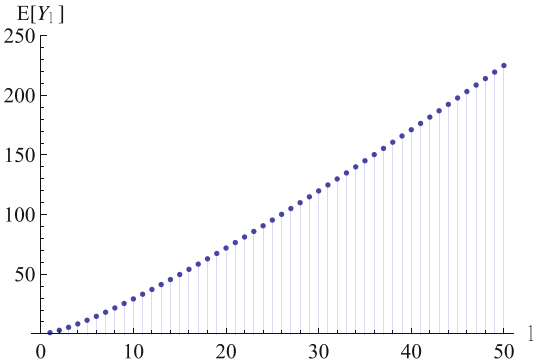

Fig. 3 The expected number $E[Y_\ell]$, where Y_ℓ is the number of random items that have to be retrieved until all ℓ target items have been found

Lemma 2 *The average number μ of distinct random items after drawing m random target items is*

$$\mu = E[Z] = \ell(1 - (1 - 1/\ell)^m). \tag{3}$$

Proof For each distinct target item i, let Z_i be an indicator random variable such that

$$Z_i = \begin{cases} 0, & \text{if item i has not been selected after m random items,} \\ 1, & \text{if item i has been selected after m random items.} \end{cases} \tag{4}$$

Then

$$E[Z_i] = Pr[Z_i = 1] = 1 - Pr[Z_i = 0] = 1 - (1 - 1/\ell)^m. \tag{5}$$

For $Z = Z_1 + \cdots + Z_\ell$ the average number of distinct target items after m random items is $E[Z] = E[Z_1 + \cdots + Z_\ell] = \ell(1 - (1 - 1/\ell)^m)$. □

In Fig. 4, the upper line shows for $\ell = 50$ how the expected number of target items found increases with the number of random items retrieved. Again, dividing the number of random items by $r = 5$ gives the average number of scrambled queries.

The lower line in Fig. 4 presents a lower (tail) bound on the number of target items found. More precisely, the line shows that with probability at least 0.9, at least so many items have been found. The corresponding tail inequality is obtained from (Motwani and Raghavan 1995, theorem 4.18) by focusing on the number Z of occupied bins instead of the number of empty bins. Then, as in the original theorem of Motwani and Raghavan (1995), a corresponding martingale sequence[4] is defined and then Azuma's inequality is applied. The outcome is that for $\lambda > 0$,

$$Pr[|Z - \mu| \geq \lambda] \leq 2 \exp\left(-\frac{\lambda^2(\ell - 1/2)}{\ell^2 - \mu^2}\right), \tag{6}$$

where Z is the number of distinct target items found after m random target items and $\mu = E[Z]$. Setting the right hand side in the above equation to be $\leq \rho = 0.1$ and solving for λ gives that $\lambda \geq \sqrt{\frac{(\ell^2 - \mu^2)\ln(2/\rho)}{\ell - 1/2}}$. The minimum possible value of λ given by the above inequality is used to draw the lower line in Fig. 4.

[4] A simple definition of a martingale sequence from Motwani and Raghavan (1995): A sequence of random variables $X_0, X_1, \ldots$, is said to be a martingale sequence if for all $i > 0, E[X_i|X_0, \ldots, X_{i-1}] = X_{i-1}$.

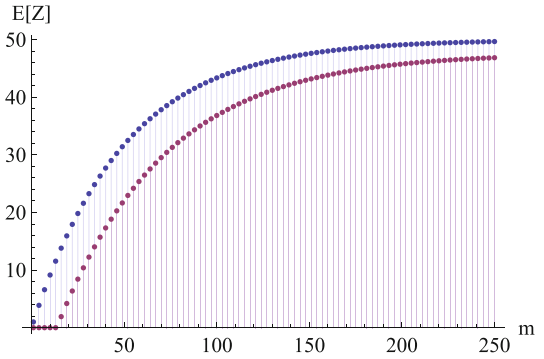

Fig. 4 The random variable Z is the number of distinct target items (out of a total of $\ell = 50$ distinct target items) that have been found after m random target items have been retrieved. For $\ell = 50$, the *upper line* shows the expected value of $E[Z]$ with respect to m. The *lower line* shows a lower (*tail*) bound on Z (with probability at least 0.9) with respect to m; with probability at least 0.9 the value of Z is not below the *lower line*

2.4 Current implementation

In order to generate scrambled queries representing hyper concepts of the query, we currently employ an ontology for natural language terms. The approach taken is a brute force one which does not involve deep semantic analysis of the query.

First, we perform a massive indiscriminate generalization taking all possible combinations of generalized query terms up to a certain higher conceptual level. Then, we apply a similarity measure to determine the distance between the query and scrambled queries; the further the distance, the better the privacy enhancement. In this respect, the similarity measure is 'loaded' with the task of classifying the scrambled queries into privacy levels, getting rid at the same time of generalized queries unsuitable to the task.

2.4.1 Query generalization

As an ontology, we employ WordNet, version 3.0 (2006), a freely available lexical database used extensively for supporting automatic text analysis and artificial intelligence applications (Miller, 1995). WordNet attempts to model the lexical knowledge of a native speaker of English. Its database contains about 150,000 words or collocations[5], organized in over 115000 synonym sets (synsets) across four types of part of speech (PoS): noun, verb, adjective, and adverb.

A synset is the smallest unit, which represents a specific meaning of a word or collocation. Synsets are connected to each other through explicit semantic relations. The hypernymy/hyponymy relations for nouns and the hypernymy/troponymy for verbs constitute an 'is-a-(kind-of)' hierarchy. The holonymy/meronymy relations for nouns constitute 'is-a-part/member/substance-of' hierarchies. Such taxonomic properties for adverbs and adjectives do not exist in the ontology. The synsets are also organized into senses.

Initially, WordNet's lemmatization process is applied to each keyword of the query, followed by stopword removal using the traditional SMART system's English stoplist.

[5] A collocation is two or more words that often go together to form a specific meaning, e.g., 'hot dog'.

Then, possible collocations are recognized by checking consequent query words against WordNet. All resulting *terms* (i.e., single keywords or collocations) go through part-of-speech (PoS) and sense disambiguation.

PoS and sense disambiguation cannot be performed well without enough contextual information, e.g., a complete sentence. Thus, we used a manual approach which gives the user more control over the whole procedure; the extra user effort is deemed insignificant in the big picture of privacy enhancement, considering also the fact that web queries consist of only 2 to 3 terms on average. The system finds all possible PoS for each term using Wordnet and prompts the user to select the proper one. Similarly, the user selects the proper sense.

Hyper-concepts for query's terms are approximated via hypernyms and holonyms for nouns, and hypernyms for verbs. For each query term, a bag of related terms is generated following the hypernymy and holonymy relations in the ontology up to a minimum level of 2 or up to 3 if level 2 results to less than 300 scrambled queries. The set of scrambled queries is the Cartesian product of those bags of words. Thus, accounting for collocations, scrambled queries have length equal to the query.

We do not generalize adverbs or adjectives since WordNet does not have similar relations, but keep them in scrambled queries. This does not seem to be a problem; adverbs and adjectives are unlikely to have privacy issues, since they are usually modifiers to verbs and nouns, respectively.

2.4.2 *Measuring privacy enhancement*

Several methods for determining semantic similarity between terms have been proposed in the literature. We apply the approach of Wu and Palmer (1994) to estimate the semantic similarity between two terms. The method has been found to be among the best edge counting methods applied on WordNet (Varelas et al. 2005), and it has been used widely in the literature, e.g., Strube and Ponzetto (2006); Yan et al. (2006). It measures the depth of the two concepts in the WordNet taxonomy as well as the depth of the least common subsumer (LCS)[6], and combines these figures into a similarity score

$$\mathrm{sim}_{i,j} = \frac{2\ \mathrm{depth}(\mathrm{LCS})}{\mathrm{depth}(i) + \mathrm{depth}(j)} \tag{7}$$

where, for the task at hand, we will denote a query term with i and a scrambled query term with j.

The similarity between pairs of terms is used to calculate the similarity between each scrambled query and the query. Let SQ be a scrambled query. If q is the length of the query, then any SQ has also length q. Thus, there are q^2 term(SQ)-to-term(query) similarities. For each scrambled query term j, what determines the privacy level is its max similarity with any of the query terms, i.e., $\max_i \mathrm{sim}_{i,j}$; the larger the max, the lesser the privacy. Similarly, for a multi-term query what determines the privacy level is the least private term, justifying again the use of max. Thus, the similarity $\mathrm{sim}_{\mathrm{SQ}}$ between the scrambled query and the query is

[6] The LCS is defined as the ancestor node common to both input synsets whose shortest path to the root node is the longest.

$$\text{sim}_{SQ} = \max_j \max_i \text{sim}_{i,j} \tag{8}$$

where $\max_j$ selects the most exposing scrambled query term with respect to the query terms.

The last measure is a very strict criterion for privacy. In the current implementation, considering that adverbs and adjectives appear in scrambled queries unchanged, the measure would return 1 denoting no privacy. In this respect, we relax the criterion by taking the average instead:

$$\text{sim}_{SQ} = \frac{1}{q}\sum_j \max_i \text{sim}_{i,j} \tag{9}$$

On the one hand, this implies that adverbs and adjectives reduce privacy, but not destroying it altogether. This reduction makes the measure safer from a privacy perspective. On the other hand, a too general term would not contribute too much to increasing the privacy of a multi-term scrambled query: too general terms are filtered out by limiting the paths on the ontology to 2 or 3 edges, as described in Sect. 2.4.1.

Table 2 shows all scrambled queries generated with the current query generalization method for the query 'gun racks', together with their similarities to the query as these are calculated by Eq. 9.

2.5 Descrambling ranked-lists

Each scrambled query run on a search engine produces a scrambled ranking. We investigate two ways of reconstructing the target ranking from many scrambled rankings.

2.5.1 Fusion

A natural and efficient approach to reconstructing the target ranking would be to fuse the scrambled rankings. However, standard fusion methods from meta-search, such as CompSUM, Borda Count, etc., may not be suitable: the scrambled rankings are results of queries targeting different, more general than the query, information needs.

Figure 2c depicts a document space, with the areas targeted by a query and two scrambled queries. The further from a query's center, the deeper in the ranking. The results we are interested in appear deeper in scrambled rankings than their top ranks. To complicate things further, web search engines usually do not return scores. Thus, a fusion approach should be based solely on ranks and have a positive bias at deep or possibly middle ranks of scrambled rankings.

A simple method that may indirectly achieve the desired result is to fuse by the number of scrambled rankings an item appears in. Assuming that sets of top results of scrambled rankings, as well as sets of noisy results, would be more disjoint than sets of deep to middle results, such a fusion method would over-weigh and rank at the top the common good results. We will call this *fusion by occurrence count* (FOC) descrambling. The method results to a rough fused ranking since it classifies items into v ranks, where v is the number of scrambled queries or rankings.

In order to determine whether Fig. 2 corresponds well to the reality of the proposed scrambler, we will also fuse with Borda Count (BC). BC is a consensus-based electoral system which in its simplest form assigns votes to ranks as $N - \text{rank} + 1$, where N is the total number of items. Since N is unknown for web search engines, we set it to 1,000, i.e.,

Table 2 All scrambled queries for the query 'gun racks'

sim_{SQ}	SQ
0.9442725	weapon system support
0.9442725	weapon support
0.9442725	arm support
0.9150327	instrument support
0.9111842	weapon system device
0.9111842	weapon device
0.9111842	arm device
0.8952206	device support
0.8819444	instrument device
0.8736842	weapon system instrumentation
0.8736842	weapon system instrumentality
0.8736842	weapon instrumentation
0.8736842	weapon instrumentality
0.8736842	arm instrumentation
0.8736842	arm instrumentality
0.8621324	device device
0.8503268	instrument instrumentation
0.8503268	instrument instrumentality
0.8433824	device instrumentation
0.8433824	device instrumentality

the depth of the scrambled lists. Then, votes per item are added for all rankings, and items are sorted in a decreasing number of total votes. Note that BC results in a smoother ranking than FOC.

2.5.2 *Local re-indexing*

Another approach to re-constructing a target ranking, which does not suffer from low correspondence of ranks to relevance and produces smoother rankings than FOC or BC, would be to recover item scores. This can be achieved by re-indexing the union of scrambled results at the user's end, and running the query against a local engine. We will call this method *local re-indexing* (LR) descrambling.

Re-indexing such non-random subsets of a web collection would locally create different frequency statistics than these at the remote end. This may result in a ranking quality inferior to the target ranking, even if all target results are found by the scrambled queries and re-indexed. Furthermore, it is inefficient compared to the fusion approaches: retrieving and indexing the union of results may introduce a significant network load, increased disk usage, and CPU load.

3 Evaluation

In order to evaluate the effectiveness of the QueryScrambler and how its quality trades off with scrambling intensity and scrambled query volume, we set up an offline experiment. We are currently not interested in the efficiency of the approach, as long as the time and

space needed is within the reaches of current commodity desktop systems and retail Internet speeds.

First, we describe the datasets, the software and parameters, and the effectiveness measures used. Then, we present the experimental results.

3.1 Datasets, tools and methods

The query dataset consisted of 95 queries selected independently by four human subjects from various query-logs. The selection was based on the rather subjective criterion of: queries which may have required some degree of privacy. Table 1 presents a sample of the test queries; the full set of the test queries is available online.[7]

The ClueWeb09 dataset consists of about 1 billion web pages, in 10 languages, crawled in January and February, 2009.[8] It was created by the Language Technologies Institute at CMU. As a document collection, we used the ClueWeb09_B dataset consisting of the first 50 million English pages of the ClueWeb09 dataset. The dataset was indexed with the Lemur Toolkit V4.11 and Indri V2.11, using the default settings of these versions, except that we enabled the Krovetz stemmer.[9] We used the baseline language model for retrieval, also with the default smoothing rules and parameters. This index and retrieval model simulate the remote web search engine.

A local engine re-indexes, per query, the union of sets of results returned by the remote engine for all scrambled queries. For the local engine, we again used the Lemur Toolkit and Indri, but in order to simulate that a remote engine's model is usually proprietary, we switched the local retrieval model to tf.idf. The items for re-indexing were extracted as term vectors directly from the remote engine's index; this implies a common pre-processing (e.g., tokenization, stemming, etc.) across the remote and local engines.

3.2 Effectiveness measures

There are several ways for measuring the top-n quality of an IR system, e.g., precision and recall at various values of n, mean average precision (MAP), etc. These compare two top-n lists by comparing them both to the ground truth, but this presents two limitations in the current setup. First, such measures typically give absolute ratings of top-n lists, rather than a relative measure of distance. Second, in the context of the web, there is often no clear notion of what ground truth is, so they are harder to use.

We are interested in the quality of the re-constructed ranking in terms of how well it approximates the target ranking, not in the degree of relevance of the re-constructed result-list. Although, this could still be measured indirectly as a percentage loss of a traditional IR measure (assuming ground-truth exists), e.g., MAP, we find more suitable to opt for direct measures of result set intersection and rank distance. In this way we will still measure the effectiveness even for queries poorly formulated for the information need, or information needs with near zero relevance in a collection.

A simple approach to measure the distance between two top-n lists τ_1, τ_2, is to regard them as sets and capture the extent of overlap between them. We measure the overlap with the following disjointness metric (DM), which is based on the symmetric difference of the two lists:

[7] http://lethe.nonrelevant.net/datasets/95-seed-queries-v1.0.txt

[8] http://boston.lti.cs.cmu.edu/Data/clueweb09/

[9] http://www.lemurproject.org

$$\mathrm{DM}(\tau_1, \tau_2) = \frac{|(\tau_1 - \tau_2) \cup (\tau_2 - \tau_1)|}{|\tau_1| + |\tau_2|}. \quad (10)$$

It lies in [0, 1], with 1 denoting disjoint lists. For lists of the same size, DM equals 1 minus the fraction of overlap.

Traditional measures of rank distance (i.e., distance between two permutations), such as Kendall's tau distance (Kendall, 1938) or Spearman's rho, are not very suitable because our lists are truncated so they may rank different results. Thus, we use *Kendall's distance with penalty parameter p*, denoted $K^{(p)}$, which is a generalization of Kendall's tau distance to the case of truncated lists. $K^{(p)}$ was introduced in Fagin et al. (2003), where it was shown that it is not a metric in the strict mathematical sense, but still a near metric in the sense of satisfying a 'relaxed' triangle inequality. On the other hand, DM is a metric.

The original Kendall distance between two permutations is essentially equal to the number of exchanges in a bubblesort to convert one permutation to the other. The generalized $\bar{K}_{i,j}^{(p)}(\tau_1, \tau_2)$ measure is also related to the permutation distance between the two truncated lists, albeit with some plausible assumptions about items that do not belong to both lists. The detailed description of $K^{(p)}$ is out of the scope of this work and the interested reader is referred to Fagin et al. (2003). In short, we first define a penalty $\bar{K}_{i,j}^{(p)}(\tau_1, \tau_2)$ for each pair of items in $P(\tau_1, \tau_2)$, where $P(\tau_1,\tau_2)$ is the union of the sets of items of the two lists. Then,

$$K^{(p)}(\tau_1, \tau_2) = \sum^{\{i,j\} \in P(\tau_1, \tau_2)} \bar{K}_{i,j}^{(p)}(\tau_1, \tau_2). \quad (11)$$

From the definition of $K^{(p)}(\tau_1,\tau_2)$ and assuming that the penalty parameter is $p \in [0, 1]$, the maximum distance between two top-k lists occurs when the lists are disjoint. In this case the value of the distance measure is $k((p + 1)k + 2 - p)$. We use the above maximum value of the $K^{(p)}$ measure to normalize it; the normalized distance takes values in the interval [0,1]. We report results with $p = 0.5$; this corresponds to the 'neutral' approach, and moreover, $K^{(0.5)}$ is equivalent to other rank distance measures (K_{avg}, $K_{\mathrm{Hausdorff}}$), as it is shown in Fagin et al. (2003).

A very important feature of the Kendall's distance with penalty parameter p is that it is a measure that can be applied even if the lists are obtained from a very large universe whose exact size might be unknown, thus it is suitable in the web retrieval context.

We evaluate with the averages of both measures on the test query dataset at top-ℓ for $\ell = 50$ instead of $n = 1000$. We find top-50 to be more realistic for web retrieval than the top-1000 of traditional IR evaluations. In addition, this allows us to put our results somewhat in perspective with the $K^{(0)}$ results for top-50 reported in Fagin et al. (2003) where rankings returned from different web search engines for the same query are compared to each other. In initial experiments, we found that $K^{(0)}$ and $K^{(0.5)}$ give values not too far away from each other. The authors in the last-mentioned study regard values of around 0.3 as 'very similar' rankings, while comparing a ranking fused from several engines to the individual rankings generated $K^{(0)}$ distances between 0.3 and 0.8.

3.3 Experiments and results

We run experiments for 3 levels of scrambling intensity and 3 levels of query volume. By looking into the sets of scrambled queries generated via the method described in Sect. 2.4, it seemed that a test query to scrambled query similarity of less than 0.70 results in

Table 3 Numbers of test queries and (min, median, max) numbers of scrambled queries per scrambling-volume combination

	Scrambling		
	Low	Med	High
Volume			
High	55 (27, 50, 50)	33 (29, 50, 50)	19 (26, 50, 50)
Med	72 (11, 25, 25)	62 (13, 25, 25)	30 (11, 25, 25)
Low	94 (3, 10, 10)	88 (1, 10, 10)	58 (1, 10, 10)

Table 4 Mean $K^{(0.5)}$ and DM for FOC descrambling

	Mean $K^{(0.5)}$			Mean DM		
	Scrambling			Scrambling		
	Low	Med	High	Low	Med	High
Volume						
High	0.980	0.989	0.998	0.985	0.992	0.999
Med	**0.961**	0.978	0.998	**0.968**	0.983	0.999
Low	0.962	0.969	0.993	0.971	0.977	0.996

extremely weak semantic relationship between the two. Consequently, we took the similarity intervals of (1, 0.7], (0.9, 0.7], and (0.8, 0.7], for low, medium, and high scrambling respectively. For scrambled query volume, we arbitrarily selected volumes in {1, 10}, {11, 25}, and {26, 50}, for low, medium, and high volume respectively.

When a combination of intensity and volume levels had 0 scrambled queries for a test query, we did not take that test query into account in averaging results. In such cases, search privacy for the query at the requested scrambling intensity and volume is not possible with the proposed method and other methods must be applied. Table 3 presents the number of test queries averaged per combination. In the parentheses, we further give the minimum, median, and maximum numbers (in this order) of scrambled queries that the test queries had for the combination at hand. The combinations with the fewest test queries are the ones where a high volume was requested, especially at high scrambling; the proposed method can generate a limited number of scrambled queries. This can be a limitation of all ontology-based methods which statistical methods may not have.

Tables 4 and 5 present the mean $K^{(0.5)}$ and DM (Sect. 3.2) for FOC and BC descrambling (Sect. 2.5.1) respectively. The best results are expected at the top-left corners of the tables for both measures, i.e., high-volume/low-scrambling, and are expected to decline with decreasing volume and/or increasing scrambling. The best experimental results are in boldface. In all experiments, the two measures appear correlated, in the sense that a better DM also implies a better ranking or $K^{(0.5)}$.

The best DM results correspond to an average intersection of only 2 or 3 results between fused and target top-50 rankings, for both fusion methods. In any case or measure, BC works better than FOC. This seems to be a result of the rougher ranking that FOC provides, since the results of the two methods become closer as volume increases. Results degrade with increasing scrambling, as expected, but also degrade with increasing volume. The

Table 5 Mean $K^{(0.5)}$ and DM for BC descrambling

	Mean $K^{(0.5)}$			Mean DM		
	Scrambling			Scrambling		
	Low	Med	High	Low	Med	High
Volume						
High	0.970	0.981	0.994	0.978	0.987	0.996
Med	0.944	0.971	0.994	0.956	0.978	0.996
Low	**0.927**	0.958	0.983	**0.944**	0.969	0.988

later is due to the fact that larger volumes of scrambled queries presuppose larger degrees of scrambling even within the same scrambling interval.

Table 6 presents results for LR descrambling (Sect. 2.5.2); they are much better than the fusion descrambling results. The unexpected degradation with increasing volume appears again, but only at low or med scrambling. However, it is now more difficult to explain, and we can only speculate that it is a result of having biased global statistics in the local collection. Here, the best DM result corresponds to an average intersection of 7 to 8 results between descrambled and target top-50 rankings.

4 Retrieval failure analysis and improvements

The task we set out to perform is daunting. Nevertheless, on average, we get to the same 7 or 8 results of the top-50 of the plain query, without submitting its important keywords; we consider this a decent result. In this section, however, we examine further the results in order to identify what may have a negative impact on retrieval effectiveness and suggest future improvements.

Firstly, it is easy to see that we have a problem in fusing scrambled ranked lists, since the Local Re-indexing and re-ranking approach (LR descrambling) yields the triple effectiveness of the two fusion methods we have tried. Nevertheless, we do not believe that LR descrambling represents the ceiling of achievable performance. In order to measure the quality of scrambled queries without the influence of descrambling, (i.e., neither fusion nor local re-indexing is used), we can look at the number of the target top-50 results found by all scrambled queries combined. Table 7 presents these numbers, averaged over all test queries. The previously best result of 7 or 8 is now raised to almost 13. We see improvements of at least 40 % and up to 100 % all over the table. In other words, although

Table 6 Mean $K^{(0.5)}$ and DM for LR descrambling

	Mean $K^{(0.5)}$			Mean DM		
	Scrambling			Scrambling		
	Low	med	high	Low	Med	High
Volume						
High	0.848	0.898	0.864	0.891	0.926	0.906
Med	0.832	0.883	0.901	0.876	0.915	0.932
Low	**0.812**	0.870	0.914	**0.856**	0.903	0.940

Table 7 Mean number of the target top-50 results found by all scrambled queries combined

	Scrambling		
	Low	Med	High
Volume			
High	11.1	9.7	7.5
Med	12.1	7.8	5.1
Low	**12.7**	8.0	4.3

the scrambled queries retrieve quite a few of the target top-50 results, local re-indexing can rank roughly half or two-thirds of those in the descrambled top-50. This is clearly due to having biased term frequency statistics in the local collection, and results could be improved by using a generic source of frequencies instead.

Secondly, there seems to be amble room for improving the method of generating scrambled queries. Let us consider a user who wants maximum privacy (i.e., high scrambling) irrespective of cost (i.e., she is willing to trade off time and use a high query volume). Assuming ideal descrambling (i.e. yielding the results of Table 7), the ceiling of performance would be 7.5 items out of 50, or a 15 % intersection between targeted and obtained results in the top-50. The 'missing' 85 % represents the price that such a user needs to pay for privacy, under the currently proposed method for generating scrambled queries. Milder privacy requirements (i.e., low scrambling) can raise the intersection to 22–25 % (or lower the missing items to 75–78 %). In any case, many target items are missed, thus we examined the recall of the scrambled queries with respect to the overall recall that they achieve for the original user query. To this end, we used the results of Sect. 2.3 to predict the overall recall from the recall of the scrambled queries and compared this number with the experimental values. Indicative results are presented in Fig. 5.

In general, the measured overall recall is lower then its predicted value. The distance becomes larger for larger volumes of random target items (which in most cases implies a larger number of scrambled queries or scrambled queries of lower scrambling degree). A plausible explanation for the divergence of the measured recall is that the target items captured by the scrambled queries are correlated and not independent samples of the set of target items as it should be in the ideal case. This in turn provides a strong indication that the scrambled queries do not catch independent random subsets of the set of target items. Instead many scrambled queries return practically the same target items in their results.

The above observation defines an important issue related to the retrieval procedure with scrambled queries. The challenge is how to select scrambled queries such that they cover more effectively the whole range of target items of the original query. The Wordnet-based approach used in this work is a first step in this direction but the results show that it can be improved. Specifically, the problem seems to be that the nearness of two terms in Wordnet's graph does not imply a high co-occurrence of them in documents. In this context, we are considering to enhance our scrambled query generation procedure with statistical methods, e.g., incorporate term co-occurrence statistics. Wordnet also presents a couple of other limitations in this context: there is no obvious way to deal with phrases (except collocations), and it is a rather generic, domain-independent, thesaurus. Domain-specific knowledge could be beneficial.

Thirdly, using a search engine based on semantics might improve results. The approach taken is based on the premise that a generalization of a concept X appears in some documents that treat X. However, most of the currently big and popular commercial search engines—as well as the research engine we used in the experiments—use very little of the

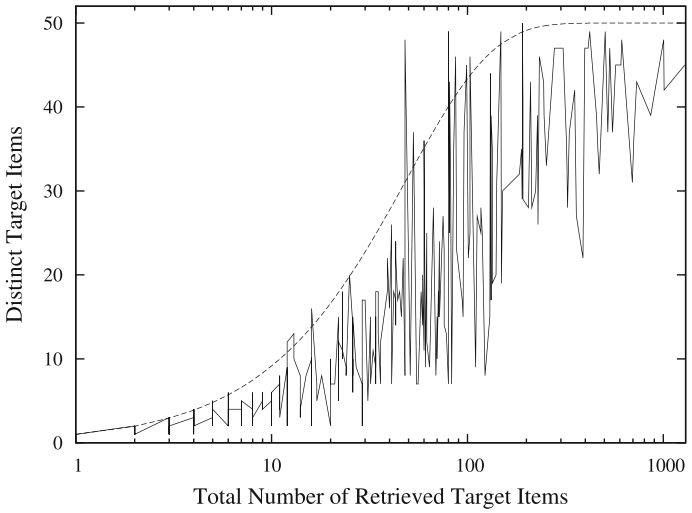

Fig. 5 The total number of distinct target items with respect to the total number of random target items for each query-experiment. The *dashed line* shows the expected number of distinct target items, if the target items are independently randomly selected. The *continuous line* presents the experimental results. The *line* simply connects a large number of points, where each point corresponds to a single experiment

semantic structure behind the concepts for ranking items. Consequently, we are falling back to simply targeting co-occurrence between user and scrambled terms, missing relevant results. To this end, we submitted a dozen of our test queries to the standard non-semantic engine of Google as well as two semantic engines: Cognition[10] and Hakia[11] Instead of the user query "acute hepatitis" (which returned 10 good results in all three engines), we submitted the scrambled query "acute liver disease". By visually examining the snippets of the top-10 results, Google returned 0 results on hepatitis, while Medline.Cognition returned 4, and Hakia (which groups results in categories) returned 1 in Credible, 3 in Pubmed, 0 in News, and 7 in Blogs. This example suggests that using semantic engines (as well as domain-knowledge) would improve results. Nevertheless, we had a difficult time finding another so good example, thus it is unclear how big the benefits may currently be. Although there are risks in exchanging a popular big and proven non-semantic search engine with an unproven semantic engine of possibly less coverage, this matter certainly deserves a further investigation in the current context.

Lastly, our results may be more promising when measured in absolute retrieval effectiveness than goodness of approximating the target ranking. We set our ultimate goal to reconstructing the target ranking, i.e., the one the original query would have returned. However, by submitting multiple scrambled queries, we may retrieve new relevant documents not appearing in a target top-50 ranking. However, we cannot measure this in our current testbed, since it consists of custom queries with privacy issues for which no relevance judgements exist. This may even be 'tricky' in other standard testbeds: Given that it is customary to approximate ground-truths through pooling processes (i.e., humans judging only the union of top results of many different systems assuming all the rest non-relevant), and that most systems participating in a pool are based on the bag-of-words

[10] http://www.congition.com

[11] http://www.hakia.com

paradigm, the ground-truth provided with standard Web test collections may not be sufficient for our purpose. Again, further experimentation with semantic engines seems important.

5 Privacy analysis

In the previous section we identified parts of our approach which may have a negative impact on retrieval effectiveness. Here, after a brief discussion on IR privacy, we investigate whether our approach achieves its the privacy goals.

5.1 Discussion on IR privacy

An interesting concept that fits into the context of privacy enhanced web search is *plausible deniability*, a legal concept which refers to the lack of evidence proving an allegation. The scrambler may enhance the plausible deniability, since the original query is never disclosed and the real interest of the user is hidden within a broader concept space. This is the main privacy-enhancing feature of the QueryScrambler. Instead of the user query, a set of scrambled queries representing more general concepts is used.

Also of great importance is the fact that the QueryScrambler can perform searches while protecting not only the privacy of the user but also the query itself. A query may pose serious privacy threats even if it is submitted anonymously. In competitive fields like business or academic research, a query may contain some interesting new idea which should not be disclosed at least while the user is still making background searches on it.

In addition to the original query, the privacy of a web search might also be endangered by the results of the query. If the results have very high precision, then important information about the original user interest might be inferred from them. Regarding this issue, an inherent privacy-enhancing principle of the QueryScrambler is that each scrambled query usually retrieves only a small number of items that are in the real interest of the user. The results of an ideal scrambled query should contain some of the target items but only mixed with a large volume of unrelated or loosely related items and preferably not in the first ranks. More specifically, each scrambled query should have small recall and small precision with respect to the ℓ most related target items. In this way, the privacy of the user is not seriously endangered by simply monitoring the results. A decisive parameter of the QueryScrambler that can be used to achieve this is the degree of scrambling. If the scrambling degree is too high, then the recall in the scrambled results will be too low; privacy will be preserved, but the query will remain unanswered. If the scrambling degree is too low, then the final recall will be high but the user privacy will be endangered.

A formal criterion that can be used for privacy is *k-anonymity* (Sweeney 2002), which demands that every piece of information about items in a set be indistinguishably related to no fewer than k items. There are more than one ways this concept can be used with the QSP. One is the approach used in Domingo-Ferrer et al. (2009b) to hide a term of the plain query within a group of k terms. As noted earlier, the weakness of this approach is that in this case k is bounded by a very small constant number. A more robust approach would be to achieve k-anonymity or k-indistinguishability in the concept space of a search query. The higher-level scrambled query should indistinguishably correspond to a large number k of conceptually lower-level terms. This way, the real interest of the user is hidden within a large field. This is our aim, and as we show in Sect. 5.2, on average the QueryScrambler

can accomplish this goal. However, there is still work to be done. For example, there are particular cases where the semantic generalization applied by the QueryScrambler may achieve k-anonymity only for small values of k. This happens when a higher-level term corresponds only to a small number of lower-level terms. We will further investigate this issue and examine ways to overcome it, for example, by employing additional sources of semantic information or extending our approach with statistical techniques.

5.2 Privacy of the QueryScrambler

The privacy goal of the QueryScrambler is to protect the actual interest of a user who wants to submit a query to a search engine. We assume that the user's interest is expressed with the original query, thus, this query should not be disclosed. We also assume that the ground-truth with respect to the retrieval task is given with the results (in this work the top-50 items) of the search engine for the original query. Consequently, these results should also be protected. In summary, we define the following privacy requirements:

A. The original user query should not be disclosed.
B. The results of the original query must be protected.

A common way to examine the effectiveness of security or privacy measures is to evaluate the system against a so-called *adversary*, who (in the QSP context) represents the malicious entity whose aim is to violate the privacy of the user by identifying her true search interest. What are the features of the adversary? If an adversary can monitor all scrambled queries, then we expect that the privacy of the plain query can be violated. However, as we discussed earlier, we can encounter the possibility of such an attack; for example, the query scrambling can be combined with other approaches like submitting individual queries via different agents or through the Tor network etc. If the adversary captures only individual scrambled queries, then the privacy of the user depends on the minimum distance between the scrambled queries and the plain query. A different attack is to use the search results of a scrambled query to extract information about the interest. As noted earlier, the QueryScrambler has the potential to generate scrambled queries of low-enough precision and high-enough recall to unbrace such attacks. We assume that an adversary

– knows that a particular scrambled query is actually scrambled,
– can capture any scrambled query but cannot link independent scrambled queries to the same original query,
– can also capture the results that are returned by the search engine for any particular scrambled query, and
– has no background knowledge about the original query.

Next, we consider the criterion of k-anonymity, or more appropriately in this context, k-indistinguishability for the user's interest as a plausible criterion for the privacy goals of the QueryScrambler. An adversary should not be able to come closer to the real interest of the user than a set of k possible interests.

Requirement A, is addressed with the query scrambling procedure described in Sect. 2.4. In the experiments, each scrambled term can correspond to any of each descendant nodes in the 2 or 3 lower levels of the Wordnet hierarchy. In our query test-set there were on average about 319 distinct words in the two lower levels. Any of these words could be the original query term that gave the corresponding scrambled query term that was

intercepted by the adversary. For example, the original term 'cortisone'[12], gives the scrambled term 'hormone'[13] (in this case a scrambled term three levels higher than the original term), and the indistinguishability for cortisone is 1 out of 138, i.e., $k = 138$. For multi term queries, indistinguishability increases multiplicatively with each additional query term. Thus, on average, every original query is indistinguishably hidden within a large number of possible terms.

An important issue is that, given a scrambled query, the corresponding candidate seed queries may not all be equally plausible. However, this non-uniformity of the candidate seed query set does not mean that an adversary who is attacking the system can with certainty exclude some of them. Let us consider the following example: An adversary who is aware of the QueryScrambling approach intercepts the scrambled query "manufacturer portable device", and then calculates a corresponding (large) set of candidate seed queries. Assume that the queries "Nokia tablet"[14] and "Apple Tablet" both belong to this set. Can the adversary exclude one of them, for example "Nokia tablet"? In our view, the fact that the query about Apple is more likely does not necessarily mean that the user did not submit the other one. In fact, the less common candidate seed query can even be more interesting, because it might reveal some interest about a less expected topic. Concluding, despite the fact that some of the candidate seed queries may be more likely than others, in most cases it should not be possible to significantly reduce the set of candidate queries using this information. Combining this with the large average number of candidate seed queries per scrambled query we believe that indistinguishability holds for the QueryScrambler approach.

Requirement B, is addressed implicitly by keeping the recall and the precision of each scrambled query low as discussed above. Even though the low precision of the scrambled queries seems to be an unavoidable consequence of the query scrambling procedure, it suits very well this requirement. In our experiments we measured an average precision of 0.0053 (or 0.018 if we exclude scrambled queries that completely failed, i.e., zero precision) at the top-1000 results. Even if an adversary knew this average precision, the low precision makes the relevant result item indistinguishable within the 1000 returned items. For example, if there are 6 relevant items, then each item of the scrambled results is a real item with probability 0.006 and even worse the correct set of the 6 relevant item is indistinguishable from a total of $C(1000, 6) \simeq 1.37 \cdot 10^{15}$ combinations (here subsets) of 1000 items taken 6 at a time. Of course, the remaining 994 items may also convey some information about the user query but they do not belong to the assumed ground-truth for the query.

In our view, the above arguments indicate that a good scrambled query can force an adversary to examine a prohibitive large set of possible interests of the user. Thus, on the one hand, we achieve the privacy goals. On the other hand, the high level of privacy for several queries in our test-bed may also justify the low retrieval effectiveness achieved. Concluding, further investigation is needed to strike a better, more usable, balance between privacy and retrieval effectiveness.

6 Conclusions

We introduced a method for search privacy on the Internet, which is orthogonal to standard methods such as using anonymized connections, agents, obfuscating by random additional

[12] More precisely, "cortisone#n#1" in Wordnet, i.e., its 1st, most-frequent, sense as a noun.

[13] More precisely, "hormone#n#1" in Wordnet, i.e., its 1st, most-frequent, sense as a noun.

[14] At the time of the writing of this paper, there was no Nokia tablet in the market.

queries or added keywords, and other techniques preventing private information leakage. The method enhances plausible deniability against query-logs by employing semantically more general queries for the intended information need. The key assumption is: the more general a concept is, the less private information it conveys; an assumption deemed true by example. We theoretically modelled the problem, providing a framework on which similar approaches may be built in the future.

The current implementation is based on a semantic ontology without using sophisticated natural language processing techniques or deep semantic analysis. It is arguably a brute force approach focusing on investigating the practical feasibility of the proposed method and the trade-off between quality of retrieved results and privacy enhancement. The proposed scrambling method gets up to 25 % of the top-50 target results, in the ceiling of its performance. Obviously, there is a price to pay for privacy, i.e., a retrieval effectiveness loss. We investigated this trade-off in a system study; it should also be investigated in a user study in order to determine the levels of trade-off users find acceptable. Overall, the exercise demonstrated promising aspects and revealed important issues that future research should tackle.

There seems to be room for improving the method of generating scrambled queries. A thorough study of query transitions, from which one might be able to take ideas for improving the scrambled queries, is in Boldi et al. (2009). Also, knowledge of user behavior (Spink et al. 2001) could help to improve such privacy protocols. Most importantly, the failure analysis suggested that using semantic engines, as well as domain-knowledge, would improve results. Another direction to pursue is the fusion of loosely-related data such as results corresponding to queries targeting different but related topics. This may have further extensions for meta-search, or ad hoc retrieval via multiple queries. Also, it seems interesting to investigate the retrieval effectiveness on non-uniform collection samples such as samples obtained via related queries. We have merely scratched the surface of a series of interesting aspects which beyond enhancing privacy may also prove useful for improving retrieval.

A complete scrambler-based system for privacy-preserving Internet search could be as follows. The steps to obtain a set of scrambled queries for an original user query can be executed locally at the user's side. The scrambled queries can then be submitted to search engines or any appropriate information providers. This step should not reveal the IP of the user. Furthermore, the scrambled queries should not be linkable with each other, thus, the interaction with search engines should not leak any information that might link the scrambled queries. Existing tools like Tor and OptimizedGoogle Search show how this can be done. Results are de-scrambled locally. An important feature of the proposed method is that it can be deployed in the current Internet; there are no requirements or assumptions on current search engines and, moreover, there is no need for external trusted parties or other external parties at all.

Acknowledgments We thank Jaap Kamps from University of Amsterdam, the Netherlands, for providing access to the ClueWeb09_B dataset, and Savvas Chatzichristofis from Democritus University of Thrace for creating Figs. 1 and 2.

References

Barbaro, M., & Zeller, T. (2006). *A face is exposed for AOL searcher no. 4417749*. Accessed June 3, 2010 from http://www.nytimes.com/2006/08/09/technology/09aol.html.

Boldi, P., Bonchi, F., Castillo, C., & Vigna, S. (2009). From "Dango" to "Japanese Cakes": Query reformulation models and patterns. In: *WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM*

International Joint Conference on Web Intelligence and Intelligent Agent Technology (pp.183–190). Washington, DC, USA: IEEE Computer Society.

Chor, B., Gilboa, N., & Naor, M. (1997). *Private information retrieval by keywords. Tech. Rep. Technical Report TR CS0917*. Haifa: Department of Computer Science, Technion, Israel Institute of Technology.

Domingo-Ferrer, J., Bras-Amorós, M., Wu, Q., & Manjón, J. A. (2009a). User-private information retrieval based on a peer-to-peer community. *Data & Knowledge Engineering 68*(11), 1237–1252.

Domingo-Ferrer, J., Solanas, A., & Castella-Roca, J. (2009b). h(k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review, 33*(4), 720–744.

Erola, A., Castellà-Roca, J., Navarro-Arribas, G., & c Torra, V. (2011). Semantic microaggregation for the anonymization of query logs using the open directory project. *SORT—Statistics and Operations Research Transactions*, 41–58).

Fagin, R., Kumar, R., & Sivakumar, D. (2003). Comparing top k lists. *SIAM Journal on Discrete Mathematics, 17*(1), 134–160.

Howe, D. C., & Nissenbaum, H. (2009). TrackMeNot: Resisting surveillance in web search. In: *Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society* (Chap. 23, pp. 417–436). Oxford, UK: Oxford University Press.

Jones, R., Kumar, R., Pang, B., & Tomkins, A. (2008). Vanity fair: Privacy in querylog bundles. In: *CIKM '08: Proceeding of the 17th ACM Conference on Information and Knowledge Management* (pp. 853–862). New York, NY, USA: ACM.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika, 30*(1/2), 81–93.

Kumar, R., Novak, J., Pang, B., & Tomkins, A. (2007). On anonymizing query logs via token-based hashing. In: *WWW '07: Proceedings of the 16th International Conference on World Wide Web* (pp. 629–638). New York, NY, USA: ACM.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM, 38*(1), 39–41.

Mitzenmacher, M., & Upfal, E. (2005). *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge, MA: Cambridge University Press.

Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge, MA: Cambridge University Press.

Murugesan, M., & Clifton, C. (2009). Providing privacy through plausibly deniable search. In: *SDM, SIAM* (pp. 768–779).

Ostrovsky, R., & Skeith, W. I. (2007). A survey of single-database PIR: techniques and applications. In: *Public Key Cryptography (PKC 2007), Lecture Notes in Computer Science*. (Vol. 4450, pp. 393–411). Berlin and Heidelberg:Springer.

Pang, H., Ding, X., & Xiao, X. (2010). Embellishing text search queries to protect user privacy. *Proceedings of the VLDB Endowment, 3*(1), 598–607.

Pass, G., Chowdhury, A., & Torgeson, C. (2006). A picture of search. In: *InfoScale '06: Proceedings of the 1st International Conference on Scalable Information Systems*. New York, NY, USA: ACM Press.

Raykova, M., Vo, B., Bellovin, S. M., & Malkin, T. (2009). Secure anonymous database search. In: R. Sion & D. Song (Eds.), *CCSW, ACM*, pp. 115–126.

Saint-Jean, F., Johnson, A., Boneh, D., & Feigenbaum, J. (2007). Private web search. In: *WPES '07: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*. (pp. 84–90). New York, NY, USA: ACM.

Shen, X., Tan, B., & Zhai, C. (2007). Privacy protection in personalized search. *SIGIR Forum, 41*(1), 4–17.

Spink, A., Wolfram, D., Jansen, M. B. J., & Saracevic, T. (2001). Searching the web: The public and their queries. *Journal of the American Society for Information Science and Technology, 52*(3), 226–234.

Strube, M., & Ponzetto, S. P. (2006). Wikirelate! computing semantic relatedness using wikipedia. In: *Proceedings of the 21st National Conference on Artificial Intelligence*. (Vol. 2, pp 1419–1424). Menlo Park, CA:AAAI Press.

Sweeney, L. (2002). k-Anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10*(5), 557–570.

Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E. G., & Milios, E. E. (2005). Semantic similarity methods in wordnet and their application to information retrieval on the web. In: *WIDM '05: Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*. (pp. 10–16). New York, NY, USA: ACM.

Wu, Z., & Palmer, M. (1994). Verb semantics and lexical selection. In: *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics* (pp. 113–138). New Mexico: Las Cruces.

Yan, P., Jiao, Y., Hurson, A. R., & Potok, T. E. (2006). Semantic-based information retrieval of biomedical data. In: *SAC '06: Proceedings of the 2006 ACM Symposium on Applied computing* (pp. 1700–1704). New York, NY, USA: ACM.

Yekhanin, S. (2010). Private information retrieval. *Communications of the ACM, 53*(4), 68–73.