Enhancing Deniability against Query-Logs

Avi Arampatzis, Pavlos Efraimidis, and George Drosatos

Department of Electrical and Computer Engineering,
Democritus University of Thrace, Xanthi 67100, Greece
{avi,pefraimi,gdrosato}@ee.duth.gr

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.

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 provider or any other site in the network path. Table 2 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 related ongoing research on web-log anonymization, where the use of fairly advanced techniques like token-base hashing [7] and query-log bundling [6] shows that the problem is by far not solved. 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 three months [11]. 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 [1]. 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 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 [5] 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 combination of anonymization tools is employed in the Private Web Search tool [12], which is also available as an (outdated) Firefox add-on.

An interesting approach was presented in [3], where a single-term query is mixed with a set of $k-1$ random 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 [3] it raises practical issues; the number of terms in a search query is bounded by a small number, 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 [8] 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 [13].

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.

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 [18,9]. An interesting approach for private information retrieval that combines homomorphic encryption with the embellishment of user queries with decoy terms is presented in [10]. However, all these 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 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*.

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 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.

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.

2 A Query Scrambler

The proposed QueryScrambler is based on a semantic framework (Section 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.

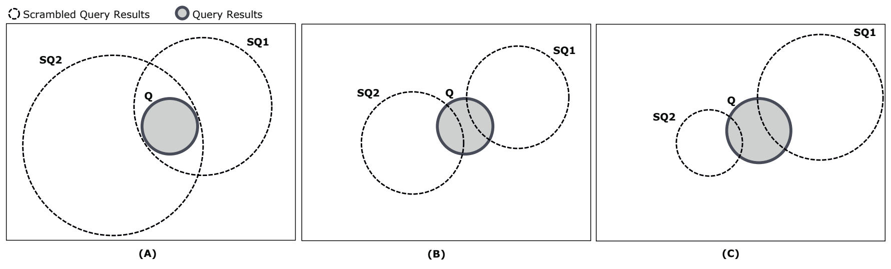

Fig. 1. Results for two scrambled queries in relation to a query Q: (A) all results in a concept space of uniform density, (B) top-n results in a uniform document space, (C) top-n results in a non-uniform document space. Q represents all relevant results.

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 1A 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 assume that the space has a uniform concept density. Both SQ1 and SQ2 cover Q completely.

Trying to transform Figure 1A to a document space, some important issues come at play. First, 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 fulfill Figure 1A. Second, web search engines usually do not return the full set of results but truncate at some rank n. Third, 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 Figure 1B.

Factoring in non-uniformity of the document space, i.e., the third issue, the picture changes to Figure 1C; 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.

2.3 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.

Query Generalization. As an ontology, we employ WordNet version 3.0 (2006). 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. 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.

Measuring Privacy Enhancement. Several methods for determining semantic similarity between terms have been proposed in the literature. We apply the approach of Wu & Palmer [16] to estimate the semantic similarity between two terms. The method has been found to be among the best edge counting methods applied on WordNet [15], and it has been used widely in the literature, e.g. [14,17]. It measures the depth of the two concepts in the WordNet taxonomy as well as the depth of the least common subsumer (LCS), and combines these figures into a similarity score $\mathrm{sim}_{i,j}$, 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 $\mathrm{sim}_{\mathrm{SQ}} = \max_j \max_i \mathrm{sim}_{i,j}$, 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:

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

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 scrambled query term would not artificially enhance too much 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.

Table 1 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 Equation 1.

2.4 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.

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,

Table 1. All scrambled queries for the query ‘gun racks’

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

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 1C 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 Figure 1 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 1000, i.e., 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. Nevertheless, both approaches suffer from the low correspondence of ranks to relevance.

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. 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 test queries were handpicked from real queries of the AOL query-log [11]. The full AOL dataset consists of 21 million queries from the AOL search (March–May 2006). Four human subjects independently selected a total of 95 queries which, in their opinion, may have required some degree of privacy. Table 2 presents a sample of the test queries; we will make their full set available online.

The ClueWeb09 dataset consists of about 1 billion web pages, in 10 languages, crawled in January and February, 2009. It was created by the Language Technologies Institute at CMU. It can be considered compatible with the test query set, since one of the many methods used to develop the ClueWeb09 employed queries sampled from the AOL query-log. 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. 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.

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.

Table 2. A sample of the 95 queries with possible privacy issues, handpicked from the AOL log

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

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 intersection metric (IM), which is based on the symmetric difference of the two lists: IM$(\tau_1, \tau_2) = |(\tau_1 - \tau_2) \cup (\tau_2 - \tau_1)\ |\ /(|\tau_1| + |\tau_2|)$. It lies in $[0, 1]$, with 1 denoting disjoint lists. For lists of the same size, IM equals 1 minus the fraction of overlap.

Traditional measures of rank distance (i.e., distance between two permutations), such as Kendall's tau distance 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 [4], 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, IM is a metric. 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 very a 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 [4] 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 $K^{(0.5)}$ get 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.

3.2 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 Section 2.3, it seemed that a test query to scrambled query similarity of less than 0.70 results in 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.

Where 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

Table 3. # of test queries and (min, median, max) # of scrambled queries per scrambling/volume

		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 IM for FOC

		mean $K^{(0.5)}$			mean IM		
		scrambling			scrambling		
		low	med	high	low	med	high
volume	high	.980	.989	.998	.985	.992	.999
	med	**.961**	.978	.998	**.968**	.983	.999
	low	.962	.969	.993	.971	.977	.996

Table 5. Mean $K^{(0.5)}$ and IM for BC

		mean $K^{(0.5)}$			mean IM		
		scrambling			scrambling		
		low	med	high	low	med	high
volume	high	.970	.981	.994	.978	.987	.996
	med	.944	.971	.994	.956	.978	.996
	low	**.927**	.958	.983	**.944**	.969	.988

further give the minimum, median, and maximum numbers 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 IM (Section 3.1) for FOC and BC descrambling (Section 2.4) 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 IM also implies a better ranking or $K^{(0.5)}$.

The best IM 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 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 (Section 2.4); 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 IM result corresponds to an average intersection of 7 to 8 results between descrambled and target top-50 rankings.

The task we set out to perform is daunting. Nevertheless, 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 principle, we may be uncovering relevant documents which do not contain any of the keywords of the plain query. However, this is difficult to measure without having the absolute ground-truth.

Table 6. Mean $K^{(0.5)}$ and IM for LR

		mean $K^{(0.5)}$			mean IM		
		scrambling			scrambling		
		low	med	high	low	med	high
volume	high	.848	.898	.864	.891	.926	.906
	med	.832	.883	.901	.876	.915	.932
	low	**.812**	.870	.914	**.856**	.903	.940

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

		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

In order to measure the quality of scrambled queries without the influence of descrambling, 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 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 can easily be improved by using a generic source of frequencies instead.

4 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 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 modeled 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, at 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 [2]. 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. We have merely scratched the surface of a series of interesting aspects which beyond enhancing privacy may also prove useful for improving retrieval.

Acknowledgments

We thank J. Kamps for providing access to ClueWeb09_B, and S. Chatzichristofis for creating Figure 1. This research has received funding from the E.U. 7th Framework Programme [FP7 2007-2013] under grant agreement no 264226: SPace Internetworking CEnter—SPICE.

References

1. Barbaro, M., Zeller, T.: A Face Is Exposed for AOL Searcher No. 4417749 (2006), `http://www.nytimes.com/2006/08/09/technology/09aol.html` (accessed June 2010)
2. Boldi, P., Bonchi, F., Castillo, C., Vigna, S.: From "Dango" to "Japanese Cakes": Query reformulation models and patterns. In: WI-IAT, pp. 183–190. IEEE Computer Society, Los Alamitos (2009)
3. Domingo-Ferrer, J., Solanas, A., Castella-Roca, J.: h(k)-private information retrieval from privacy-uncooperative queryable databases. Online Inf. Review 33(4), 720–744 (2009)
4. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. SIAM J. Discrete Math. 17(1), 134–160 (2003)
5. Howe, D.C., Nissenbaum, H.: TrackMeNot: Resisting surveillance in web search. In: Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society, ch. 23, pp. 417–436. Oxford University Press, Oxford (2009)
6. Jones, R., Kumar, R., Pang, B., Tomkins, A.: Vanity fair: privacy in querylog bundles. In: CIKM, pp. 853–862. ACM, New York (2008)
7. Kumar, R., Novak, J., Pang, B., Tomkins, A.: On anonymizing query logs via token-based hashing. In: WWW, pp. 629–638. ACM, New York (2007)
8. Murugesan, M., Clifton, C.: Providing privacy through plausibly deniable search. In: SDM, pp. 768–779. SIAM, Philadelphia (2009)
9. Ostrovsky, R., Skeith, W.I.: A survey of single-database PIR: techniques and applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
10. Pang, H., Ding, X., Xiao, X.: Embellishing text search queries to protect user privacy. In: VLDB (2010)
11. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: InfoScale. ACM, New York (2006)
12. Saint-Jean, F., Johnson, A., Boneh, D., Feigenbaum, J.: Private web search. In: WPES, pp. 84–90. ACM, New York (2007)
13. Shen, X., Tan, B., Zhai, C.: Privacy protection in personalized search. SIGIR Forum 41(1), 4–17 (2007)
14. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: AAAI, pp. 1419–1424. AAAI Press, Menlo Park (2006)
15. Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E., Milios, E.: Semantic similarity methods in wordnet and their application to information retrieval on the web. In: WIDM, p. 16. ACM, New York (2005)
16. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: Proc. of the 32nd Ann. Meeting of the Assoc. for Computational Linguistics, Las Cruces, New Mexico, pp. 133–138 (1994)
17. Yan, P., Jiao, Y., Hurson, A., Potok, T.: Semantic-based information retrieval of biomedical data. In: SAC, p. 1704. ACM, New York (2006)
18. Yekhanin, S.: Private information retrieval. Commun. ACM 53(4), 68–73 (2010)