A Versatile Tool for Privacy-Enhanced Web Search

Avi Arampatzis, George Drosatos, and Pavlos S. Efraimidis

Department of Electrical and Computer Engineering
Democritus University of Thrace, Xanthi 67 100, Greece
{avi,gdrosato,pefraimi}@ee.duth.gr

Abstract. We consider the problem of privacy leaks suffered by Internet users when they perform web searches, and propose a framework to mitigate them. Our approach, which builds upon and improves recent work on search privacy, approximates the target search results by replacing the private user query with a set of blurred or scrambled queries. The results of the scrambled queries are then used to cover the original user interest. We model the problem theoretically, define a set of privacy objectives with respect to web search and investigate the effectiveness of the proposed solution with a set of real queries on a large web collection. Experiments show great improvements in retrieval effectiveness over a previously reported baseline in the literature. Furthermore, the methods are more versatile, predictably-behaved, applicable to a wider range of information needs, and the privacy they provide is more comprehensible to the end-user.

1 Introduction

In 2006, AOL released query-log data containing about 21 million web queries collected from about 650,000 users over three months [8]. To protect user privacy, each real IP address had been replaced with a random ID. Soon after the release, the first 'anonymous' user had been identified from the data [2]. Interestingly, this identification was made solely on the queries attributed to an anonymous 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.

The incident has motivated lots of research in web-log anonymization and solutions using anonymized or encrypted connections, agents, obfuscating by random additional queries, and other techniques; for a recent extensive review on the literature, we refer the reader to [1]. There is an important reason why all the aforementioned methods alone might be inadequate: in all cases, the query is revealed in its clear form. Thus, such approaches 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 towards the *existence of a private search task* at the user's end is weakened. In other words, when a user employs the above technologies, the engine still knows that *someone* is looking for "lawyers for victims of child rape", and the user cannot deny that she has a private search task which may be the aforementioned one.

A way to achieve plausible deniability was recently presented in [1], called *query scrambler*, and works as follows. Given a private query, generate a set of *scrambled*

P. Serdyukov et al. (Eds.): ECIR 2013, LNCS 7814, pp. 368–379, 2013.

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 result-lists called *scrambled rankings*. Given the scrambled rankings, it is attempted to reconstruct, at the searcher's end, a ranking similar to the one that the private query would have produced, called *target ranking*. The process of reconstruction is called *descrambling*. The scrambler employed semantically more general queries for the private query, by using WordNet's ontology. The key assumption was: the more general a concept is, the less private information it conveys. Addressing privacy issues has the inherent difficulty to define what privacy really means. Privacy is an elusive concept, encompassing different things in different contexts and for different people [9].

The main contributions of this work are the following. In contrast to the semantic framework used in previous work, we employ a purely statistical framework. Within this statistical framework, we define three comprehensive privacy objectives—including the equivalent of the privacy objective introduced in [1]. These objectives are used to define and quantify the privacy guarantees for a given web search task. All statistics needed for generating scrambled queries are estimated on a query-based document sample of the remote engine [5]; consequently, the tools presented in this work are corpus-specific. Compared to the semantic approach, our methods are found to be significantly better in retrieval effectiveness, better defined, more versatile, predictably behaved, applicable to a wider range of information needs, and the privacy they provide is more comprehensible to the end-user.

2 A Statistical Approach to Query Scrambling

We assume an Internet user with an information need expressed as a query for a public web search engine like Google, Bing or Baidu. The retrieval task we focus on is document discovery, i.e. finding documents that fulfill the information need of the user.

The Query Scrambling Problem (QSP) [1] for privacy-preserving web search is defined as: Given a private query q for a web search, it is requested to obtain the related web documents as if q had been submitted to a search engine. To achieve this, it is allowed to interact with search engines, but without revealing q; 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 q should be kept as low as possible.

Given a private query q, we identify two types of privacy-sensitive resources:

- The q itself and the corresponding information need of the user. In this work, we use q and information need interchangeably.
- The document set matching q, given by a public search engine. An adversary monitoring these results can extract significant information about the information need.

We will define two privacy primitives for web-search. Let N be the size of the document collection, H_q the set of documents matching q, and $\mathrm{df}_q = |H_q|$ the document frequency of q. Finally, let $\mathrm{df}_{w,q} = |H_w \cap H_q|$, for any query w and q. Let us, for now, imagine that w and q are single-term queries, so H_w and H_q are determined simply by the document sets their terms occur in; in Secs. 3.1 and 3.2 we will see how we deal with multi-term queries.

A popular privacy primitive is *k-anonymity* [10], or *k-indistinghuishability*, which in the context of our work means that an adversary should not be able to come closer than a set of k possible alternatives to the private resource. Given q, for a candidate scrambled query w the first primitive k_w is

$$k_w = \frac{\mathrm{df}_w}{\mathrm{df}_{w,q}} \, , \tag{1}$$

a privacy measure between the two queries based on the concept of k-indistinguishability of the results. Note, that k_w is the inverse precision of the retrieval results of w with respect to the results of q. From a privacy perspective, submitting w instead of q, each of q's target documents is 'hidden' within at least $k_w - 1$ other documents.

The second primitive g_w is

$$g_w = \frac{\mathrm{df}_w}{N} \, , \tag{2}$$

a measure of the generality of w. The rationale behind g_w is that a general query can be assumed to be less exposing. As an indication of how general a query is, we use a pure statistical measure: *The more documents of the collection a query hits, the more general the query is.*

Based on the above primitives we define the following privacy objectives and present a use-case for each of them:

- Anything-But-This privacy or ABT_k: Assume a researcher in academia or industry who is working on some new application or product. The researcher might be interested in searching about her new idea, but might hesitate to submit a query in a clear form to a public search engine. Additionally, she doesn't care about what else will be revealed as long as it isn't her true interest. With ABT_k the researcher can conduct a scrambled search where each scrambled query satisfies $k_w > k$.
- Relative-Generalization privacy or RG_r: A citizen might be looking for information about some disease, but would not like to disclose the exact disease. A scrambled search based on scrambled queries more general than q by a factor of r might serve her need, while significantly reducing her privacy risks. Formally, RG_r means that every w must satisfy $g_w > r \cdot g_q$.
- Absolute-Generalization privacy or AG_g: Consider a citizen in some totalitarian regime. The user might decide to scramble one or more sensitive queries, for example about specific human rights, into queries with generality above a given user-specified threshold. In this case, every scrambled query must satisfy $g_w > g$.

These three privacy types may be combined, if such a privacy request arises. We will not investigate such scenarios in this paper. Note that the minimum RG privacy (RG_1) also assures the minimum ABT privacy (ABT_1) but not the other way around.

Clearly, in realistic settings, it is not be feasible to calculate the exact values of the privacy measures defined above, since no one but the engine itself has access to its full collection. However, we can resort to estimating the needed quantities from a query-based document sample of the engine.

We can now model our query scrambling approach as a set covering problem [6]. More precisely, we define Scrambled Set Covering $SSC(v, k, g)$, a multi-objective extension of set covering. Given a finite universe U of all documents of a collection, a partition of U into sets H_q and $U - H_q$, and a collection S of subsets of U, the requirement is to find a subset C of S to satisfy the following objectives and/or constraints:

- maximize $(\bigcup_{H_w \in C} H_w) \bigcap H_q$, i.e., to maximize the coverage of H_q,
- $|C| \leq v$, where v is the maximum number of scrambled queries,
- for each $H_w \in C$, the corresponding scrambled query w must satisfy $k_w > k$,
- for each $H_w \in C$, the corresponding scrambled query w must satisfy $g_w > g$.

For example, the SSC instance $SSC(10, 2, 0.01)$ refers to query with 10 scrambled queries, ABT_2 and $\mathrm{AG}_{0.01}$. The same example with RG_2 would be $SSC(10, 2, 2\,g_q)$.

Let us give an overview of our approach for query scrambling. First, we obtain a collection sample of size N with a query-based document sampling tool; this is done offline, however, the sample should be updated often enough to correspond to significant collection updates at the remote engine. In the online phase:

1. A private query q is decomposed into a set of scrambled queries. The scrambled queries are chosen to satisfy the user-specified privacy objectives of Sec. 2. To this end, we employ statistical information from the collection sample.
2. The scrambled queries are submitted as independent searches and all results are collected. To avoid a reverse engineering attack, the scrambled queries should not be linkable to each other. The user should use Tor or other anonymization tools for the submissions, taking care to assure unlinkability between the scrambled queries.
3. The query q may be locally executed on the scrambled results (local re-indexing), or the scrambled ranked-lists may be fused with some combination method.

The tool we propose is intended to be used in the following way: A user can install it locally and then use it to scramble privacy-sensitive queries. It does not rely on some trusted third party for the scrambling process.

3 Generating Scrambled Queries

For generating scrambled queries, we follow a statistical approach using a local document sample of the remote search engine. So far, for simplicity, we have assumed single-term private and scrambled queries. In Secs. 3.1 and 3.2, we will see how we can generalize the methods to work with multi-term queries. As soon as we generate a set of candidate scrambled queries, these are filtered for privacy according to the measures defined in Sec. 2. The remaining candidates are ranked according to their expected retrieval effectiveness, described in Sec. 3.3, before they are submitted.

3.1 Dealing with Multi-term Private Queries

If q is a single-term query, then its document frequency df_q can be determined directly from the document sample. The question is how to treat a multi-term q, or else, what the df_q of such a query is and which subset of df_q documents will be assumed as matching q so we can harvest from it related terms to be used as scrambled queries.

Given df_q, the question of which subset of documents is matching q can be settled as: we rank the sample documents with respect to q using some best-match retrieval model and ORed q, and take the top-df_q documents. We determine the threshold df_q by submitting the ANDed q to the collection sample and count the number of results, enforcing a minimum of 1 for practical reasons. We will refer to this estimate of df_q as aDF. The maximum number of results an ANDed query can retrieve is $\min_i \mathrm{df}_i$ when i is a term of the query; we will refer to such an estimate of df_q as mDF. This happens when the query term with the least df is 100% positively correlated with all other query terms. The term with the least df is also the most informative: if we were to reduce a multi-term q to a single term, this is the term we would keep. In this respect, df_q cannot be larger than mDF in any case.

While aDF may be too restrictive especially for a long q, mDF may be too 'loose' especially if q contains high frequency common terms. So, we will employ both aDF and mDF for estimating df_q. From a retrieval perspective, it is easier to create scrambled queries to retrieve smaller sets of documents, thus, using aDF makes the task easier than using mDF. From a privacy perspective, mDF is the largest df possible so it is safer. For example let us consider the information need represented by the query "big bad wolf". Using aDF will point to documents about the "Little Red Riding Hood" fairy tale, while using mDF will point to all documents referring to wolves including the fairy tale. Since aDF's target set is smaller, it can be easier retrieved by scrambled queries. But using mDF instead corresponds to trying to hide all wolves.

3.2 Generating Multi-term Scrambled Queries

For single-term scrambled queries, df_w can be determined directly from the document sample. However, we can also generate multi-word scrambled queries. The question is how to treat these, or else, what the df_w of such a scrambled query is and which subset of df_w sample documents will be assumed as occurring in.

From the documents matching q, we enrich the set of candidate scrambled single-term queries by using a sliding window of length W and generating all unique unordered combinations of 2 and 3 terms. We use a window instead of whole documents so as to limit the number of combinations; currently, we set $W = 16$ which was shown in past literature to perform best in ensuring some relatedness between terms [11] (see also Sec. 3.3). We limit the scrambled query length to 3, which also helps to keep the number of combinations practically manageable. In this procedure, we exclude all stopwords except those occurring in q.

The document set hit by such a scrambled query is estimated similarly to the method of aDF described in Sec. 3.1: The ORed scrambled query is submitted to the sample and the top-df_w documents are considered matching, where df_w is the number of documents matching the ANDed scrambled query. The choice of aDF over mDF is made purely on targeting the best privacy. aDF produces lower df_w estimates than mDF, so these queries will be removed earlier as g increases. Also, using aDF implies that queries are more targeted, achieving higher precision, so they will be removed earlier as k increases.

3.3 Ranking Scrambled Queries

After dropping candidate scrambled queries that violate any privacy criteria on k_w and g_w, the remaining queries should be ranked according to their expected retrieval quality with respect to the document set matching the query, i.e. the target set. For example, we can measure this quality in terms of precision and recall, and combine those in one number such as the F_β-measure [7]. Although F_β is suitable for our purpose, it has not been commonly used before for detecting the best related terms.

Topically-related terms can be ranked via several methods; a common one is by computing pointwise mutual information (PMI) using large co-occurrence windows [4]. For the task at hand, it is appropriate to consider whole documents as windows, and score each w co-occurring with q as

$$\mathrm{PMI}_w = \log \frac{P(q,w)}{P(q)P(w)} = \log N \frac{\mathrm{df}_{q,w}}{\mathrm{df}_q \, \mathrm{df}_w} \tag{3}$$

where $P(q,w)$ is the probability of q and w co-occurring in a document, and $P(q)$, $P(w)$, the probabilities of occurrence of q, w, in a document, respectively. Using a large corpus and human-oriented tests, [11] did a comprehensive study of a dozen word similarity measures and co-occurrence estimates. From all combinations of estimates and measures, document retrieval with a maximum window of 16 words and PMI (run tagged DR-PMI16) performed best on average.

Although PMI has been widely used in computational linguistics literature, classification, and elsewhere, it has a major drawback in our task. Removing constant factors from Eq. 3, which do not affect the relative ranking of terms for a given q and collection, PMI ranks terms identically to the ratio: $\mathrm{df}_{q,w}/\mathrm{df}_w$. Considering this ratio, an $1/1$ term is ranked higher than a $9/10$ term although the latter is clearly a better term from a retrieval perspective leading to a better recall; moreover, the former may be some accidental/spurious match. Or else, the PMI of perfectly correlated terms is higher when the combination is less frequent. This low-frequency bias may not undesirable for some tasks (e.g. collocation extraction), but it is in our case due to our high precision *and* recall preference. A workaround is instead to use a normalized version of PMI such as NPMI [3], which divides PMI by $-\log P(q,w)$, reducing some of the low frequency bias but not all. In any case, our task—while related—is not exactly a linguistic similarity one, where PMI works well in finding synonyms for TOEFL synonym tests [11], or collocation identification, where NPMI works well [3].

Our task seems more related to scoring features for feature selection in classification. [12] review feature selection methods and their impact on classification effectiveness. They find that PMI (which confusingly they refer to as just MI) is not competitive with other methods, and that the best methods are the χ^2-statistic and the expected mutual information (MI) [7, Ch. 13.5.1, Eq. 13.17] (which they refer to as information gain) with similar effectiveness. Still, our task is different than a straightforward term selection for classification. In classification, all selected terms are intended to be used simultaneously in order to classify a new object. Here, we use selected terms as queries *one by one* in order to cover the target set of documents. Beyond query volume, other parameters such as the number of documents retrieved per related query and the cardinality of the target document set may impact the effectiveness of the procedure.

All in all, since our task is different than determining linguistic similarity or feature selection, it makes sense to evaluate again some common term similarity measures and feature selection methods, as well as some uncommon ones, in this context.

4 Evaluation

In order to evaluate the effectiveness of the scrambler and how its retrieval quality trades off with scrambled query volume (v) and scrambling intensity (k or g) over the different privacy types (ABT/RG/AG) and methods (aDF/mDF), we set up an offline experiment. For comparison purposes, we re-constructed the set-up of [1] as close as possible.

4.1 Datasets, Tools and Methods

The private query dataset is available online[1] and consists of 95 queries selected independently by four human subjects from various query-logs [1]. 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, Indri V5.2, using the default settings, 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.

We took a document sample of the remote collection using random queries similarly to [5]. We bootstrapped the procedure with the initial query "www". At each step, the procedure retrieves the first K results of the random query and adds them to the sample; we set $K = 1$. Previous research has shown that the choice of the initial query is not important and that $K = 1$ is best suited for heterogeneous collections such as the web. Then, a term is uniformly selected from the unique terms of the current sample and used as the next random query until the desired sample size is reached. Candidate terms are at least 3 characters long and cannot be numbers. After initial experiments we decided to use a sample of 5,000 documents which provides a good compromise between effectiveness and practical feasibility. We used the same types of indexing and retrieval model for the sample as for the remote engine.

In initial experiments we compared PMI, NPMI, MI, F_1, F_2 and centroid weight, and found that MI and centroid weight work best for the task of ranking scrambled queries. F_β with $\beta = 2$, i.e. weighing recall twice to precision, is slightly behind but competitive; the F-measure however requires an extra parameter (β). NPMI works better than PMI, but both are left quite behind. We will not present these results for space reasons, and will stick with MI.

We targeted the top-50 documents of the remote engine. Our local sample was so small in relation to the engine's collection that all target documents corresponded to less than 1 document in the sample. In this respect, in order to improve the focus of the scrambled queries, it makes sense to harvest those from a set of sample documents of a smaller cardinality than df_q. In initial experiments we found that a good compromise between focus and reasonably good statistics of document frequencies is to take the top-df'_q sample documents returned by q, where $\mathrm{df}'_q = \min(10, \mathrm{df}_q)$, i.e. we harvested scrambled queries from the *at most* top-10 sample documents. Also, we adjusted df'_w and $\mathrm{df}'_{q,w}$ to the new set and calculated MI using these numbers instead; this was found to improve retrieval effectiveness. Of course, the privacy constraints were applied to the unmodified frequencies as described in Sec. 2.

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

Concerning the evaluation measures, we simplified the matters in relation to [1] where scrambled rankings were fused via several combination methods and the fused ranking was evaluated against the target one via Kendall's τ and a set intersection metric. The fusion methods tried in the previous study were deemed weak in comparison to a local re-indexing approach, i.e. index locally the union of top-1000 documents retrieved by all scrambled queries and run the private query against the local index in order to re-construct the target ranking. Nevertheless, even with local re-indexing the ceiling of achievable performance was not reached: there were quite a few target documents retrieved by scrambled queries that could not be locally ranked in the top-50. This was attributed to having biased DF statistics in the local index. The experimental effort in the aforementioned study concluded with a bare experiment evaluating only the number of target top-50 documents found by the union of the top-1000 documents retrieved by all scrambled queries. This allowed to remove the effect of de-scrambling and evaluate only the quality of scrambling; this is what we will also do.

4.2 Results

The two left-most columns of Table 1, marked as 'unfiltered', show results with no privacy; these can be considered as the ceiling of achievable performance when decomposing a user query q with the current methods. Even with no privacy, we do not get 50 out of 50 target documents because there are cases where we cannot exactly reproduce q from the sample for the following reasons. First, a term of q may not occur in the sample, e.g. 'chamblee' from "definition of chamblee cancer". However, such a term may occur in the remote collection. Second, the terms of a multi-term q, e.g. 'definition', 'chamblee', and 'cancer', may not occur within a window of 16 terms in sample documents. Third, we generate scrambled queries only up to 3 terms. All these already suggest future improvements: use larger samples, use larger or no windows at all but whole documents, and generate longer scrambled queries.

Table 1. ABT privacy, top-50 target documents found by the top-v scrambled queries

	unfiltered		$k = 1$		$k = 2$		$k = 4$		$k = 8$		$k = 16$	
v	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
2	30.1	34.2	28.6	30.3	19.9	12.1	11.8	5.05	7.49	2.02	3.57	1.13
10	36.2	39.6	35.3	37.5	30.9	23.6	22.8	11.2	15.6	5.13	8.33	2.41
50	40.8	44.2	40.2	42.5	37.3	33.1	31.8	19.3	23.7	10.8	14.2	5.25

Table 1 also shows results for ABT privacy. The minimum privacy ($k = 1$) removes only scrambled queries which occur in all documents of the sample target set. This has a larger impact to a single-term q which may loose its 50 out 50 effectiveness. The table also shows that for light or no privacy requirements mDF works better than aDF; this happens because the sample target set of mDF is larger than this of aDF, so more scrambled queries are harvested/generated leading to better results. However, the effectiveness of mDF degrades faster than aDF as k increases, so aDF works better, as expected and explained in Sec. 3.1. For large k (e.g. for $k \geq 2$), the effectiveness

Table 2. Top-50 target documents found, RG privacy (left), AG privacy (right)

	$g = g_q$		$g = 2\,g_q$		$g = 4\,g_q$		$g = 8\,g_q$	
v	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
2	22.1	13.5	19.9	8.17	12.6	4.33	7.35	1.65
10	31.1	21.2	31.4	12.6	22.1	6.83	13.3	3.42
50	38.3	28.6	36.1	19.2	28.9	10.3	20.1	6.28

	$g = .0064$		$g = .0128$		$g = .0256$		$g = .0512$	
v	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
2	13.7	5.22	13.5	6.82	9.29	5.40	7.80	4.83
10	21.2	11.4	21.8	11.8	15.9	12.3	11.7	7.79
50	28.0	17.1	26.5	19.0	23.1	18.0	16.0	11.4
#q	69	27	82	44	87	63	94	81

of mDF roughly halves for every doubling of k, suggesting a linear relation in log-log space or a power-law.

Table 2 shows results for RG (left) and AG privacy (right). Using mDF, RG effectiveness roughly halves for every doubling of generalization, suggesting again a power-law. Concerning AG privacy, the g values shown correspond to document frequency cut-offs of 32, 64, 128 and 256 in the current sample size. If a private query is already general enough for a g value, it is not scrambled since it has no privacy issues. Such queries are excluded from the average results of the right table. The numbers of private queries scrambled per g value and choice of aDF/mDF are shown in the last row (#q). The effectiveness of mDF is similar for the first three small g cut-offs but then falls off. In other words, we can generalize private queries relatively well by using scrambled queries hitting up to 2.5% sample documents. At such an AG level, 66% (63 out of 95) of the private query dataset is deemed as not general enough so it is scrambled. Again, the aDF method is much better than mDF in all cases, providing a less steep decrease in effectiveness as generalization increases.

The fact that aDF is more effective than mDF in all privacy types when more than light privacy is required, does not mean that it should be the preferred method. As we noted in Sec. 3.1, mDF represents stricter privacy than aDF which is experimentally proved to trade off with retrieval effectiveness. The final choice between aDF/mDF should be left to the end-user or determined via a user-study.

Concerning scrambled query volume, in all privacy types and methods effectiveness increases with higher volumes. However, due to the nature of the experimental setup, we see diminishing returns as effectiveness gets closer to 50 documents. At high privacy levels where effectiveness suffers, we can see roughly a doubling of effectiveness for every fivefold increase in volume, i.e. another power-law albeit a very steep one suggesting that a few dozens of scrambled queries are enough.

4.3 A Comparison to Semantic Query Scrambling

The previous literature dealt only with RG privacy, so we will compare our RG method and results to it. The best effectiveness reported by [1] is 12.7, obtained at low volume (i.e. as many scrambled queries as can be produced up to 10) and low scrambling by averaging the results for 94 of the 95 user queries. One query did not produce any scrambled queries at low scrambling. At higher volume, ironically, effectiveness slightly decreased, an effect we attribute to averaging only the 55 user queries having numbers of low-scrambled queries in the 26–50 range. Effectiveness decreased fast—below 10 and even 5 documents—at medium or high scrambling.

The most obvious problems of the semantic approach are the following. First, not all user queries can be scrambled at a requested scrambling intensity, due to WordNet's ontology being generic thus not 'dense' enough. The problem seems severe: at high scrambling, only 58 out of the 95 user queries had at least 1 scrambled query. Second, the levels of low/medium/high scrambling were defined by taking arbitrary ranges of values of some semantic similarity measure between each scrambled query and q. Thus, scrambling intensity is difficult to be explained to the end-user: how much exposing is a scrambled query with, say, 0.8 similarity to q?

Our statistical approach does not have the problems of the semantic. First, we always seem to produce enough scrambled queries. This may not be the case for very small document samples, but it does hold for our—reasonably small—5,000 sample. Second, our approach to RG can easier be explained to the end-user: the information need expressed by a scrambled query is satisfied by at least X times more documents than her private query. This can give her a better idea on how much she is exposed, in contrast to giving her a raw similarity threshold as in the semantic approach.

Moreover, we seem to get much better effectiveness. Although the two approaches are not directly comparable due to the weak definitions of low/medium/high scrambling of the semantic approach, comparing the methods at minimum scrambling (i.e. low scrambling vs. $g = g_q$) at volume 10 we see improvements of +145% or +67% (12.7 vs. 31.1 with aDF or 21.2 with mDF. Nevertheless, we should investigate which levels of privacy are roughly comparable across the two approaches.

Table 3. Top-10 RG scrambled queries for private query 'gun racks' and # of target docs found

low scrambling	medium scrambling	mDF, $g = g_q$	mDF, $g = 2g_q$	aDF, $g = 8g_q$
weapon system support	device support	light replacement	air power	air power
weapon support	instrument device	gun light **39**	light power	light power
arm support	weapon system instrumentation	air book cover	weight	weight
instrument support	weapon system instrumentality	electric light machine	accessory	accessory
weapon system device	weapon instrumentation	pull	machine power	machine power
weapon device	weapon instrumentality	air kit	light supply	light model
arm device	arm instrumentation	air cover	22 light	fire light
—	arm instrumentality	air gun home **3**	cover picture	gun **40**
—	device device	light pump	light model	trailer
—	instrument instrumentation	brake	fire light	air picture
0	**0**	**39**	**0**	**40**

Let us attempt a comparison of RG at the minimum level, as well as, at levels of the statistical approach which result to around 12.7 target documents on average for volume 10, according to Table 2. For the user query "gun racks", Table 3 compares the scrambled queries resulting from the semantic approach (the two left-most columns of Table 3 are taken from a table appearing in [1]) against the scrambled queries of the statistical approach. The semantic approach is capable of generating only 7 scrambled queries at low scrambling but 10 at medium scrambling. None of the scrambled queries hit any of the target documents at any scrambling intensity. A bold number next to a query is the number of target results hit (if any), while the last row shows the number of distinct target results hit by all scrambled queries per column. The statistical approach achieves good results (above the 12.7 average) in two out of three cases. Nevertheless, it seems difficult to decide where the methods stand privacy-wise: is "weapon support" less exposing than "gun light" or just "gun"? In our opinion, the user should have the last word on this by reviewing the set of scrambled queries before submission.

All in all, using the strictest privacy provided by mDF, we roughly matched or improved the best retrieval result of the semantic approach, for k up to 4 and g up to $2g_q$ or 2.5% at volume 10, and for k up to 8 and g up to $4g_q$ or 5% at volume 50. At lighter privacy requirements, we outperformed the semantic approach by far. In all cases, our methods managed to scramble all private queries where this was needed, in contrast to the semantic approach. Moreover, we detected power-law relations between the privacy levels and retrieval effectiveness of ABT and AG, as well as, between volume and retrieval effectiveness. Thus, our methods are more well-defined and easier explained to the end-user, can be applied to a wider-range of private information needs, are more effective and behave predictably, retrieval-wise.

Last, there are two other advantages of our approach over the semantic one. First, in the semantic approach the user had to manually select the part-of-speech and sense of every term in her query in order to select the right node in WordNet's ontology. The statistical approach does not require these time-consuming steps. Second, [1] arrived at the conclusion that the best method to de-scramble ranked-lists is to locally re-index the union of documents hit by all scrambled queries and run q against this local index. Nevertheless, even with local re-indexing the ceiling of achievable performance was not reached: there were quite a few target documents retrieved by scrambled queries that could not be locally ranked in the top-50. This was attributed to having biased DF statistics in the local index due to the fact that the local documents represented a far from uniform collection sample: they were all retrieved by a set of semantically-related scrambled queries. The document sample used by our approach is more representative of the remote collection, so its DF statistics can be used in the local re-indexing approach removing most of the bias.

5 Conclusion

We introduced a method for search privacy on the Internet, which is orthogonal to—and should be combined with— standard methods such as using anonymized connections, agents, obfuscating by random additional queries or added keywords, and other techniques reducing private information leakage. The method enhances plausible deniability towards query-logs by employing alternative less-exposing queries for a private query. We defined and modeled theoretically three types of privacy, providing a framework on which similar approaches may be built in the future.

In contrast to previous literature, we followed a statistical approach which does not use word/concept ontologies, semantic analysis or natural language processing. We investigated the practical feasibility of the proposed method and the trade-off between quality of retrieved results and privacy enhancement. In [1], the best result was 25% of the top-50 target documents found, and was achieved at the lightest possible privacy requirements; our method can match this at higher-than-minimum privacy levels and for more and better-defined privacy types which can easier be explained to the end-user.

At our lightest privacy level, our method outperforms the semantic one by far; we retrieve up to 56–76% of the target results. Moreover, our method can be applied to a wider range of information needs and performs more predictably retrieval-wise.

Privacy is an elusive concept. While it is easy to evaluate the retrieval effectiveness of our methods, it is difficult to evaluate the actual privacy perceived by the end users. We investigated our approach in a system-study; it should also be investigated in a user-study in order to determine the levels of privacy trade-offs users find acceptable.

Acknowledgments. The research leading to these results has received funding from the E.U. 7th Framework Programme [FP7/2007-2013] under grant agreement no 264226: SPace Internetworking CEnter—SPICE. This paper reflects only the views of the authors—The Union is not liable for any use that may be made of the information contained.

References

1. Arampatzis, A., Efraimidis, P., Drosatos, G.: Enhancing Deniability against Query-Logs. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 117–128. Springer, Heidelberg (2011)
2. 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 3, 2010)
3. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. In: GSCL, pp. 31–40. Gunter Narr Verlag, Tbingen (2009)
4. Brown, P.F., Pietra, V.J.D., de Souza, P.V., Lai, J.C., Mercer, R.L.: Class-based n-gram models of natural language. Computational Linguistics 18(4), 467–479 (1992)
5. Callan, J.P., Connell, M.E.: Query-based sampling of text databases. ACM Trans. Inf. Syst. 19(2), 97–130 (2001)
6. Caprara, A., Toth, P., Fischetti, M.: Algorithms for the set covering problem. Annals of Operations Research 98, 353–371 (2000)
7. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press (2008)
8. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: InfoScale. ACM (2006)
9. Solove, D.J.: Understanding Privacy. Harvard University Press (2008)
10. Sweeney, L.: k-Anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10(5), 557–570 (2002)
11. Terra, E.L., Clarke, C.L.A.: Frequency estimates for statistical word similarity measures. In: NAACL-HLT, pp. 165–172. ACL (2003)
12. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: ICML, pp. 412–420. Morgan Kaufmann (1997)