Inf Retrieval J
DOI 10.1007/s10791-015-9256-0

Versatile Query Scrambling for Private Web Search

Avi Arampatzis[1] · **George Drosatos**[1] · **Pavlos S. Efraimidis**[1]

Received: 25 June 2014 / Accepted: 27 April 2015

Abstract We consider the problem of privacy leaks suffered by Internet users when they perform web searches, and propose a framework to mitigate them. In brief, given a 'sensitive' search query, the objective of our work is to retrieve the target documents from a search engine without disclosing the actual query. 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 private 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 queries with privacy issues 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. Additionally, we investigate the perceived privacy via a user study, as well as, measure the system's usefulness taking into account the trade off between retrieval effectiveness and privacy. The practical feasibility of the methods is demonstrated in a field experiment, scrambling queries against a popular web search engine. The findings may have implications for other IR research areas, such as query expansion, query decomposition, and distributed retrieval.

Keywords Query scrambler · Search privacy · Query-based document sampling · Mutual information · Set covering · Inter-user agreement

An early shorter version of this work was published in European Conference on Information Retrieval, 2013.

✉ Avi Arampatzis
avi@ee.duth.gr

George Drosatos
gdrosato@ee.duth.gr

Pavlos S. Efraimidis
pefraimi@ee.duth.gr

[1] Department of Electrical and Computer Engineering, Democritus University of Thrace, University Campus, 67100 Xanthi, Greece

Published online: 16 May 2015

1 Introduction

In 2006, AOL released query-log data containing about 21 million web queries collected from about 650,000 users over 3 months (Pass et al. 2006). To protect user privacy, each IP address had been replaced with a random ID. Soon after the release, the first 'anonymous' user had been identified from the data: the user assigned the ID 4417749 was identified as the 62-old Thelma (Barbaro and Zeller 2006 (accessed June 5, 2014)). Interestingly, this identification was made solely on the queries attributed to her 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 AOL personal data breach has motivated lots of research in web-log anonymization Carpineto and Romano (2013) and solutions using anonymized or encrypted connections, agents, obfuscating by random additional queries (Murugesan and Clifton 2009) or added keywords (Domingo-Ferrer et al. 2009), and other techniques (Shen et al. 2007). Another way to perform privacy-preserving searching is to employ cryptographic tools, as in Cao et al. (2014) or Boneh and Waters (2007). However, the cryptography-based techniques presented in these papers are not applicable in the context of this work, since they rely on collaborative search engines.

Popular search engines use query-logs to build user profiles, which can then be used to offer personalized search services to the users Hannak et al. (2013). However, the user profiles built by search engines can pose serious threats against the privacy of the users. Sometimes, it is even possible to infer personal information, for example certain demographic data, from seemingly irrelevant personal data of the users Bhagat et al. (2014). Therefore, many works in the literature focus on controlling the personal information that flows into the profiles built by search engines or on obfuscating these profiles by submitting fake queries.

A popular tool that permits users to obfuscate their web-search profile is the TrackMeNot add-on (Howe and Nissenbaum 2009) for the Firefox browser. TrackMeNot tries to obfuscate the profile of a user 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. 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 (Saint-Jean et al. 2007). In Sánchez et al. (2013); Viejo and Sánchez (2014) fake search queries are used to distort in a disciplined way the user profiles built by search engines.

A community-based approach to support user privacy in information retrieval is presented in Domingo-Ferrer et al. (2009), where a user gets her query submitted by other users of a peer-to-peer community. The main idea is distort user profiles by having search queries submitted by other users. A similar approach is presented in Castellà-Roca et al. (2009), and is further extended in Lindell and Waisbard (2010) to handle malicious adversaries.

Finally, there are search engines such as DuckDuckGo[1] and Ixquick[2] that state that they do not collect any personal user information, and some relay services like Startpage[3]—and until recently Scroogle—that submit search queries to Google on behalf of anonymous

[1] https://duckduckgo.com/.

[2] https://ixquick.com/.

[3] https://www.startpage.com/.

users. For a recent extensive review on the literature, we refer the reader to Arampatzis et al. (2013) and Peddinti and Saxena (2014).

There is an important reason why the above 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[4] 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. Additionally, there are cases of web searches where the search query should not be disclosed. For example, the search may concern some new research or business idea. Finally, for very sensitive web searches, avoiding the disclosure of the actual search query may be the only safe approach. Noteworthily, it has been shown very recently, that the use of anonymization tools like Tor[5] cannot always assure that a web search will not be associated with the user who submitted the query. More precisely, Peddinti and Saxena (2014) demonstrated that an adversarial search engine, equipped with only a short-term history of a users search queries, can break the privacy guarantees of Tor and TrackMeNot by only utilizing off-the-shelf machine learning techniques.

In this work, we consider how to perform web searches without disclosing the actual search query and the corresponding search interest of the user to any party, including the web search engine. We call this the Query Scrambling Problem (QSP). A potential solution to the QSP must work with existing web search engines, assure the privacy of the web search query (and not necessarily the profile of the user at the search engine), and at the same time manage to retrieve useful results (retrieval effectiveness). The retrieval effectiveness of any method proposed for the QSP is a fundamental requirement. By definition the ground truth are the results that a web search engine would return had the private search query been submitted to it. Thus, the scrambled or blurred or whatever queries replace the private search query must retrieve as many as possible of the target results of the private query.

A way to achieve the above—which has been proven a rather ambitious goal—was introduced in Arampatzis et al. (2011), called *query scrambler*, and works as follows. Given a private query, 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 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.

A semantic approach to generate "blurred" queries is also presented in Sánchez et al. (2013). However, there the focus is on obfuscating in a controlled way the user profiles built by search engines. There is no requirement for retrieval effectiveness of the "blurred" queries, and consequently no experiments about the retrieval effectiveness are presented.

[4] *Plausible deniability* is a legal concept which refers to the lack of evidence proving an allegation. See also http://en.wikipedia.org/wiki/Plausible_deniability.

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

Dealing with the QSP problem, i.e., with how to find the target results for a particular search query without disclosing the query, is a plausible and interesting problem in the field of privacy-enhanced web searching. Nevertheless, we are not aware of related works beyond the above that address this problem or some close variation of it. Most related work either focuses on query-logs or on user profiles built by search engines. Approaches that focus on query-logs do not fit the QSP problem, since in the QSP the query should not be disclosed in the first place, and moreover, in these approaches there is usually no requirement for retrieval effectiveness. In the case of user profiles, the common practice is to submit the search query either in a way that is not linkable to the user or hidden within a set of fake queries. Such approaches can be used to protect the user profile but generally do not fit the QSP problem, since the QSP requires that the search query must not be disclosed at all.

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 Arampatzis et al. (2011). 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 (Callan and Connell 2001); we also provide some empirical heuristics for taking better samples faster. Additionally, we investigate an alternative approach based on set covering, aiming to diversify the selection of scrambled queries in the sense that each one covers a different part of the private information need. 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. Furthermore, we conduct a user study which confirms that our word-frequency based criteria can provide strong indicators to what users perceive as private or 'safe'. Last, via a field experiment with a real web search engine, we identify some technical challenges in putting the proposed methods into operation.

The rest of this paper is structured as follows. In Sect. 2 we present our approach for privacy-enhanced web search. The creation of document samples of search engines is described in Sect. 3, and the production of scrambled search queries in Sect. 4. We evaluate our methods with an extensive set of experiments in Sect. 5. Moreover, we present a set covering approach for query scrambling (Sect. 6), a user study on how actual users perceive the privacy concepts and criteria that we propose (Sect. 7), and conduct a field experiment of query scrambling on a real web search engine (Sect. 8). Conclusions are drawn in Sect. 9.

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) (Arampatzis et al. 2011) 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.

2.1 Privacy in web search

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

- The q itself representing the 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.

Some definitions are in order. Let N be the size of the document collection, H_q the set of documents matching q, and $\mathrm{df_q} = |\mathrm{H_q}|$ the document frequency of q, i.e. the number of documents in the collection that q hits. Finally, let $\mathrm{df_{w,q}} = |\mathrm{H_w} \cap \mathrm{H_q}|$, for any queries w and q. Let us, for now, assume 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 Sects. 4.1 and 4.2 we will see how we deal with multi-term queries.

We can now define two privacy primitives for web-search. The first one is based on the popular *k-anonymity* (Sweeney 2002), or *k-indistinguishability*, concept, 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. In other words, k_w gives the number of documents that each target document is hidden within, in the set of results hit by scrambled query w; the larger its k_w, the more private w is. 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, when submitting w instead of q, each of q's target documents is 'hidden' within $k_w - 1$ other documents.

The second primitive g_w is

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

a measure of the generality of w. In other words, g_w is the fraction of the collection that a scrambled query w hits. 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 w satisfies $k_w > k$, where k is a parameter (or 'knob') given to the user, with $k \geq 1$, which controls the required privacy; the larger the k, the better the privacy.
- Relative-Generalization privacy or RG_r: a person 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 $g_w > r \cdot g_q$, where $r \geq 1$. Again, r is a 'knob' for the user to decide the privacy level required; the larger the r, the better the privacy.

– 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 g. In this case, every scrambled query must satisfy $g_w > g$. As with the first two objectives, g is a user-controlled parameter in $[0, 1)$; the larger the g, the better the privacy.

These three privacy types may be combined, if such a privacy request arises. Indeed, the query scrambling approach that we present, can support arbitrary combinations of the above criteria. Nevertheless, in the experimental evaluation we will focus on each privacy criterion independently, in order to be able to draw conclusions from the outcomes. Finally, note that, by their definition, the minimum RG privacy (RG_1) also assures the minimum ABT privacy (ABT_1) but not the other way around.[6]

Clearly, in realistic settings, it is not 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.

2.2 Overview of query scrambling

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 Sect. 2.1. 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, *unlinkability* between the scrambled queries should be assured; otherwise, an adversary may collect and identify the scrambled queries of a particular scrambled search, and may be able to extract significant information about the private query q. An operational scrambling system can achieve unlinkability by using different Tor circuits or some other privacy-preservation tool for each scrambled query submission.
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. However, the vendor of the tool may distribute document samples of popular search engines in frequent time intervals.

[6] A more general query w than the private query q will always hit more results than q thus achieving some k-indistinguishability, while a w hitting other but overlapping results than q may or may not be more general.

3 Document sampling of search engines

Our approach requires a document sample of the remote search engine. Here, we will describe how we take such a sample, based on methods previously reported in the literature.

We take a sample of the collection using random queries similarly to Callan and Connell (2001); Tigelaar and Hiemstra (2010). We have slightly modified the procedure in order to achieve higher efficiency (i.e. to increase speed and reduce network load by issuing fewer queries) and reach a 'richer' sample (i.e. a sample consisting of a larger number of unique and non-unique terms). The modified procedure works as follows.

We bootstrap the procedure with the arbitrary 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 $K = 1$. The last two aforementioned works have 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. Here, we introduce the following additional restrictions over the standard procedure described in past literature: (1) candidate terms are at least three characters long and cannot be numbers, and (2) should have a document frequency of more than 1 in the current sample, as long as the sample contains at least two documents. Our experiments with sampling a few thousand documents from the ClueWeb09_B dataset[7] have shown that not using such a document frequency cutoff results to around 5 % of the random queries hitting previously retrieved documents. The cutoff removes this inefficiency by likely discarding misspellings and unique identifiers, and it moreover produces a 'richer' sample faster by slightly reducing fixation on previously seen topics: the number of terms (both total and unique) per sample size is usually higher.

Concerning the sample size, previous research, such as e.g. Callan and Connell (2001); Tigelaar and Hiemstra (2010), has shown that 200–500 documents, depending on collection characteristics, are sufficient. Most previous research, however, evaluated samples for their vocabulary coverage and document frequency distribution with respect to the whole collection. Beyond these two features, we are also interested in the quality of term co-occurrence statistics, as it will become obvious in the following section. We are also interested in knowing whether the set covering methods we will apply using a sample (Sect. 6) transfer well to the whole collection. Consequently, we will mainly experiment with a sample of 5000 documents from our test collection, which—although it is by an order of magnitude larger than the size suggested by previous research—it may or may not be large enough for our purposes; we will empirically examine this by also experimenting with larger samples, e.g. 20,000. Even larger samples—23,000 and 50,000 documents—will be used when sampling a real web search engine in Sect. 8.

4 Generating scrambled queries

For generating scrambled queries, we follow a statistical approach using the local document sample of the remote search engine. So far, for simplifying the definition of the privacy objectives in Sect. 2.1, we have assumed single-term private and scrambled

[7] http://lemurproject.org/clueweb09.

queries. In the next two subsections, we will see how we can generalize the methods to work with multi-term private queries (Sect. 4.1) and multi-term scrambled queries (Sect. 4.2).

As soon as we generate a set of candidate scrambled queries, these are filtered for privacy according to the objectives defined in Sect. 2.1. The remaining candidates are ranked according to their expected retrieval effectiveness, described in the third subsection (Sect. 4.3), and the top-v scrambled queries are submitted to the remote engine. Algorithm 1 at the end of this section gives an overview pseudo-code for the generation method.

4.1 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. Estimating df_q can be seen as a typical rank-thresholding problem. Recent approaches on rank thresholding, such as Arampatzis et al. (2009), assume a binary document relevance to the query, fit a binary mixture of probability distributions on the total score distribution, and seek to draw a score/rank threshold that optimizes a given retrieval effectiveness measure. We instead use a simpler approach, which we describe next, that does not involve relevance or selecting a measure to optimize. This approach argues about reasonable upper and lower bounds for df_q and uses those.

The minimum number of documents hit by a query q can be found by submitting q in an ANDed fashion to the collection sample and count the number of results, enforcing a minimum of 1 for practical reasons; we will refer to this lower bound as aDF. Now, the maximum number of results an ANDed query can retrieve is $\min_i \mathrm{df}_i$, where i is a query term (i.e. the number of documents the query's least frequent term hits); we will refer to this number as mDF. This maximum number of results in an ANDed query is achieved only when all other query terms co-occur in the documents hit by the query term with the least df; in any other case, less than mDF documents are hit by the ANDed query. 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 these respects, mDF gives some upper bound. Thus, we have $\mathrm{aDF} \leq \mathrm{df}_q \leq \mathrm{mDF}$.

While aDF may be too restrictive especially for a long q, mDF may be too 'loose' especially if q does not contain a low frequency term. So, we employ and evaluate both aDF and mDF as estimators of 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 (correctly), 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 (would provide stronger privacy).

4.2 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 (Terra and Clarke 2003) (see also Sect. 4.3). We limit the scrambled query length to 3, since a typical web query is usually between 2 and 3 words, which also helps to keep the number of combinations practically manageable. In this procedure, we exclude all stopwords, using the list of stopwords from the Text Categorization Project[8], but keep a stopword if it exists in q.

The document set hit by such a scrambled query is estimated similarly to the method of aDF described in Sect. 4.1: 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 are less general and 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.

4.3 Ranking scrambled queries

After dropping candidate scrambled queries w that violate privacy criteria either on k_w or g_w, the remaining queries are ranked according to their expected retrieval quality with respect to the document set matching the private 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 (Manning et al. 2008). Although F_β seems like exactly what we need for our purpose, initial tests showed that it may not be the best choice for our task due to the difficult-to-quantify effects that a retrieval model's weighting scheme has on the terms of the chosen scrambled queries. Furthermore, since it has not been commonly used before for detecting the best related terms, we also looked for alternatives.

Topically-related terms can be ranked via several methods; a common one is by computing pointwise mutual information (PMI) using large co-occurrence windows (Brown et al. 1992). For the task at hand, it is appropriate to consider whole documents as windows; PMI scores 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, Terra and Clarke (2003) did a comprehensive study of a dozen word similarity measures and co-occurrence estimates. From all combinations of estimates

[8] http://code.google.com/p/text-categorization/.

and measures, document retrieval with a maximum window of 16 words and PMI (run tagged as DR-PMI16 in the latter cited paper) performed best on average.[9] However, both document or windows-oriented approach for frequency estimates produced similar results 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: $\mathrm{df}_{q,w}/\mathrm{df}_w$. Thus, PMI does not distinguish between terms with the same ratio but different frequencies of co-occurrence with the query (hits), although we would prefer terms with more hits to achieve recall so a lower volume of scrambled queries is used. This low-frequency bias may not be undesirable for some tasks (e.g. collocation extraction), but it is a drawback in our case due to our high precision *and* recall preference. A workaround is to use instead a normalized version of PMI such as NPMI (Bouma 2009), 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 (Terra and Clarke 2003), or collocation identification, where NPMI works well (Bouma 2009).

Our task seems more related to scoring features for feature selection in classification. Yang and Pedersen (1997) 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) (Manning et al. 2008, Chapter 13.5.1, Equation 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.

There are still other IR methods that may work in the task at hand, e.g. taking the centroid of the documents used to harvest scrambled queries and rank the scrambled queries according to their centroid weight. All in all, since our task is different than determining linguistic similarity or feature selection, it makes sense to evaluate some common term similarity measures and feature selection methods, as well as some uncommon ones, in this context. In initial experiments, we compared PMI, NPMI, MI, F_1, F_2 and the weight of the centroid (of idf-only weighted documents), 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 for the rest of the paper.

[9] "Document retrieval" meant that document frequency statistics were used (as in Eq. 3) for the terms; using collection frequencies instead requires a window-based approach for calculating term co-occurrence. In both cases, "maximum window of 16 words" meant that only pairs of co-occurring words within 16 words were considered.

Algorithm 1 Generate Scrambled Queries

Require:
Private query q
Size of sliding window W
Max length of scrambled queries ℓ
Number of the top-th sample documents returned by q
Retrieval cutoff (aDF or mDF) rco
Privacy filter (ABT, RG and AG) PR_f
Ensure:
A list of scrambled queries $scrQ$, sorted on decreasing MI

$index \leftarrow$ initiate sample
$M \leftarrow$ the number of documents from sample's $index$
$scrQ \leftarrow$ initiate the list of scrambled queries
$stopWords \leftarrow$ load the list of stopwords and remove (from this list) the stopwords that exist in q
$docIDs_{cutoff} \leftarrow$ Retrieve all documents from $index$ with rco (mDF or aDF) cutoff for the q
$\text{df}_q \leftarrow$ the number of results from $docIDs_{cutoff}$
$t_q \leftarrow$ select the minimum value from the $\min\{th, \text{df}_q\}$
$docIDs \leftarrow$ Retrieve top-t_q documents from $index$ for the q

for (every document of $docIDs$) **do**
    $docVector \leftarrow$ get the vector of document and remove the $stopWords$
    $tmp_{scrQ} \leftarrow$ generate the scrambled queries from $docVector$ with window W and max length ℓ
    for (every scrambled query of tmp_{scrQ}) **do**
        if (scrambled query not in $scrQ$) **then**
            $docIDs_{scr} \leftarrow$ Retrieve all documents from $index$ with aDF cutoff for the scrambled query
            $\text{df}_w \leftarrow$ the number of results from $docIDs_{scr}$
            $hits \leftarrow$ find the number of matches of $docIDs_{scr}$ and $docIDs$
            $hits_{cutoff} \leftarrow$ find the number of matches of $docIDs_{scr}$ and $docIDs_{cutoff}$
            if ($\text{df}_w > 0$ AND $hits > 0$) **then**
                $score \leftarrow$ calculate the MI of scrambled query with these parameters: $\text{df}_w, hits, t_q, M$
                add the scrambled query in $scrQ$ with extra values: $score, \text{df}_w, hits, hits_{cutoff}$

the required privacy filter PR_f is applied by removing scrambled queries from $scrQ$
sort the remaining scrambled queries ($scrQ$) on a decreasing MI score

5 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 scrambled query generation methods (aDF/mDF), we set up an offline experiment. For comparison purposes, we used the set-up described in Arampatzis et al. (2013).

5.1 Datasets, tools and methods

The private query dataset is available online[10] and consists 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 8 presents a sample of the test queries. As a document collection, we used the

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

ClueWeb09_B dataset consisting of the first 50 million English pages of the ClueWeb09 dataset.[11] The dataset was indexed with the Lemur Toolkit, Indri V5.2, using the default settings, except that we enabled the Krovetz stemmer.[12] 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 with the method described in Sect. 3. After initial experiments and according to the heuristic estimates of the necessary sample size made in Sect. 3, we decided to use a sample of 5000 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.

We targeted the top-50 documents of the remote engine. Since our document sample was much smaller than 1/50th of the remote collection, all target top-50 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-t_q sample documents returned by q, where $t_q = \min\{10, \mathrm{df}_q\}$, i.e. we harvested scrambled queries from the *at most* top-10 sample documents. Similarly, we defined t_w and $t_{q,w}$ for 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 Sect. 2.1.

Concerning the evaluation measures, we simplified the matters in relation to Arampatzis et al. (2013) 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 reconstruct 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.

Furthermore, in this paper, we report results as *fractions* of the top-K target documents found rather than absolute *numbers* of documents. Using fractions makes it easier to directly compare results across experiments targeting different numbers of top documents, i.e. using different values for K.

5.2 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 de-composing a user query q with the current methods. Even with no privacy, we do not get 50 out

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

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

Table 1 ABT privacy; fraction of the top-50 target documents found by the top-v scrambled queries

v	Unfiltered		$k = 1$		$k = 2$		$k = 4$		$k = 8$		$k = 16$	
	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
2	.601	.683	.571	.606	.398	.241	.237	.101	.150	.040	.071	.023
10	.724	.792	.706	.751	.619	.473	.457	.225	.312	.103	.167	.048
50	.816	.885	.803	.849	.746	.662	.637	.387	.473	.216	.285	.105

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 three terms; obviously, longer private queries cannot be re-produced. Nevertheless, we do not consider all these as problems of the method, since they are introduced by technical choices we made in order to speed up the task. Only the first reason produces some uncertainty, although the problem may be eased by simply using larger samples. The other two can be completely removed by using no windows at all but whole documents, and generate longer scrambled queries.

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 Sect. 4.1. For large k (e.g. for $k \geq 2$), the effectiveness of mDF roughly halves for every doubling of k, suggesting a linear relation in log-log space or a power-law. The effectiveness of aDF degrades slower.

Tables 2 and 3 show results for RG and AG privacy respectively. 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 of 5000 documents. If a private query is already general enough for a g value, it is not scrambled since it already complies with the privacy requirements. Such queries are excluded from the average results of Table 3. The numbers of private queries scrambled—thus contributing in the results—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 roughly 2.5 % ($g = .0256$) of the 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 Sect. 4.1, 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.

Table 2 RG privacy; fraction of the top-50 target documents found by the top-v scrambled queries

v	$g = g_q$		$g = 2\,g_q$		$g = 4\,g_q$		$g = 8\,g_q$	
	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
2	.442	.271	.397	.163	.253	.087	.147	.033
10	.623	.423	.628	.253	.441	.137	.267	.068
50	.767	.571	.721	.383	.577	.206	.402	.126
2	95	95	95	95	95	94	94	88
10	95	95	95	95	95	90	94	82
50	95	95	95	94	94	88	91	69

The bottom half of the table shows the number of test private queries contributing to the averages. As privacy requirements increase, some of the private queries may have no or not enough scrambled queries at the requested volume; these private queries cannot be scrambled, thus they are excluded. This issue can be overcome by using larger sample sizes

Table 3 AG privacy; fraction of the top-50 target documents found by the top-v scrambled queries

v	$g = .0064$		$g = .0128$		$g = .0256$		$g = .0512$	
	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
2	.273	.104	.270	.136	.186	.108	.156	.097
10	.425	.228	.436	.237	.319	.246	.233	.156
50	.560	.341	.530	.380	.462	.360	.319	.228
#q	69	27	82	44	87	63	94	81

The last row shows the number of user queries found to have privacy issues at a minimum required generality level of g for aDF and mDF

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 100 %. At high privacy levels where effectiveness suffers, we can see roughly a doubling of effectiveness for every five-fold increase in volume, i.e. another power-law albeit a very steep one, suggesting that hundreds or even thousands of scrambled queries may be needed for getting close to 100 % effectiveness.

5.3 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 Arampatzis et al. (2013) is 12.7 out of 50, i.e. 0.254, obtained at low volume (i.e. as many scrambled queries as can be produced up to ten) and low scrambling (in the next paragraph there is an explanation of the classification into low/medium/high scrambling of the semantic approach) 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 ten and even five 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 one. 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—5000 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 r 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 levels of privacy or generalization are not absolutely identical due to the arbitrary 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 % (out of 50 target documents, 12.7 found in Arampatzis et al. (2013) versus 31.1 with aDF or 21.2 with mDF found according to the results in Table 2 multiplied by 50). Nevertheless, we should investigate which levels of privacy are roughly comparable across the two approaches.

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 private user query "gun racks", Table 4 compares the

Table 4 Top-10 RG scrambled queries for private query "gun racks", number of target documents found per query (a number in bold typeface next to a query–no number if no documents found), and the number of distinct target results hit by all scrambled queries per column (numbers in the last row)

Semantic query scrambling		Statistical query scrambling		
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**

scrambled queries resulting from the semantic approach (the two left-most columns of Table 4 are taken from a similar table in Arampatzis et al. (2013)) against the scrambled queries of the statistical approach. The semantic approach is capable of generating only seven scrambled queries at low scrambling but ten 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 .0256 at volume 10, and for k up to 8 and g up to $4g_q$ or .0512 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 current 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, in Arampatzis et al. (2013) we 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.

6 A set covering approach

After having established the feasibility of our query scrambling approach and the improved results with respect to previous work, we proceed with an independent experiment that extends our work and shows strong evidence about the possibilities of further optimization. In particular, first we see the problem of ranking candidate scrambled queries as a set covering problem (Sect. 6.1). Then, in Sect. 6.2, we present an approach based on exponential weights to guide the covering algorithm of the target results and show further improvements in retrieval effectiveness.

6.1 Algorithmic foundations of query scrambling

From an algorithmic point of view, the query scrambling approach is a set covering problem (Caprara et al. 1998); there is a set of documents, the target results of the private query are a specific subset of this set, and one tries to 'cover' all the target results by retrieving sets of scrambled results. The definition of the original set covering problem follows.

Definition 1 *(Set Covering)* Given a collection S of subsets of a finite set B, a cover of B is a subset C of S, such that the union of the sets in C is equal to B that is, each item of B is covered.

In the context of this work, B is the set of the results to the user query, S is the collection of the results to scrambled queries (one set of results for each scrambled query) and $C \subset S$ is a set of scrambled results that cover the set B.

Set covering is a fundamental combinatorial problem and one of the first to be shown to be NP-complete (Karp 1972). There are various extensions of the original set covering problem. One of them is particularly interesting for this work and is called red-blue set covering (Carr et al. 2000). In red-blue set covering the items are distinguished into red and blue and the objective is to find a cover of the blue items which minimizes the number of red items covered. The formal definition of red-blue covering follows.

Definition 2 *(Red-blue set covering)* Given a finite set of 'red' elements R, a finite set of 'blue' elements B and a family $S \subset 2^{R \cup B}$, the red-blue set cover problem is to find a subfamily $C \subset S$ which covers all blue elements, but which covers the minimum possible number of red elements.

In the context of query scrambling, the blue elements would be the target results, while the red elements would be the rest of the documents in the document collection. However, the objectives of query scrambling are more subtle and cannot be completely captured with red-blue set covering. For example, a perfect solution for red-blue set covering without any red elements (if possible), would not be very attractive for query scrambling, because the target results of the seed query would be identifiable and this would reveal important information about the private query.

For the needs of query scrambling we define scrambled set covering, a multi-objective extension of set covering (Fig. 1).

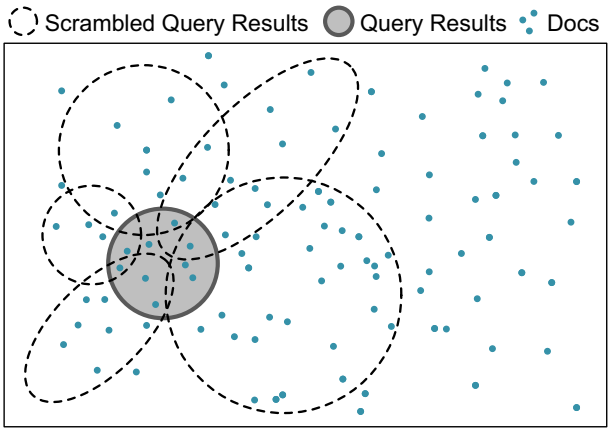

Fig. 1 Query scrambling as a set covering problem

Definition 3 Scrambled Set Covering $SSC(v, k, g)$: 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:

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

Note that the last definition supports both primitive privacy criteria, k-anonymity and generality g, and any combination of them. Consequently, any query scrambling task with one or more of the criteria ABT, RG and AG can be formulated as an instance of $SSC(v, k, g)$.

In this work, we supply bounds on the size of C, the index k and the index g, and optimize the coverage of the items in H_q. One may define other variations of SSC by switching the objective function with one of the constraints or defining an objective function comprising more than one constraints, for example minimizing a weighted sum of k and g. Moreover, scrambled set covering can be extended with further objectives, like the maximum number of times any item of H_q or any item of $U - H_q$ can appear in the cover, etc.

The computational complexity of $SSC(v, k, g)$ can easily be shown to be NP-Complete by reducing it to the original set covering problem. Moreover, the approximability of scrambled set covering with respect to objective (ii) cannot better than $\log n$ where n is the number of the target results, since this bound holds for the original set covering problem (Lund and Yannakakis 1994). The results on the approximability of red-blue set covering presented in Carr et al. (2000) do not fit the SSC problem.

6.2 A guided covering approach

We use an approach based on set covering to guide the selection of the scrambled queries. Since the scrambled set covering problem (Definition 3) is NP-Complete, we will not try to solve it optimally. Instead, we will devise a polynomial time greedy algorithm as a heuristic for the query scrambling task. Note that greedy algorithms are a popular way to deal with set covering problems (Chvatal 1979; Young 2008). Given a private query q, let $t_q = \min\{10, \mathrm{df}_q\}$. Moreover, let T_q be a set of the top t_q documents that q hits in the sample collection. Instead of choosing scrambled queries based only on the MI criterion, we will choose them incrementally while keeping track if and how many times each document in T_q has been hit so far.

We present the following greedy algorithm with exponential weights for covering the documents in T_q. Each item i in T_q has an initial weight β_i of one, and this weight is multiplied by $f = 0.5$ every time the document is covered by a scrambled query that is selected. In each round, a score for each candidate scrambled query is calculated. The scrambled query with the best score is selected and the weights of all items in T_q are updated. The process is repeated until υ scrambled queries are selected.

More precisely, let t_q and T_q be as defined above. Let C_q be the set with the incremental outcome of the algorithm. Initially, $C_q = \emptyset$. Let S_q be the set of candidate scrambled queries for q. We will run experiments for volume sizes $\upsilon = 2$, 10, and 50, as in the previous sections. We bound the size of S_q by $n = 2v_{\max}^2 + 1$, where $v_{\max}$ is the maximum

volume size. This size assures a sufficiently large pool of documents for the covering algorithm, while keeping the computational complexity of the algorithm under control.[13] Since in this set of experiments $v_{\max}$ is 50, the maximum size of S_q is $2 * 50^2 + 1 = 5001$ scrambled queries achieving the highest MI scores (see Sect. 4.3). The precision and the recall of each scrambled query are estimated with the documents frequencies (df_q, df_w and $df_{q,w}$). The initial weight of each document i in T_q is $\beta_i = 1$, and the discount factor is $f = 0.5$.

1. The following steps 2 and 3 are repeated t_q times.
2. For each scrambled query j in $S_q \setminus C_q$, (that is, the set difference $j \in S_q$ and $j \notin C_q$), calculate a score

$$\sigma_j = \left(\sum_{j \text{ hits } i} \beta_i \right) \cdot \max\{\text{Precision}_j, \text{Recall}_j\}. \tag{4}$$

3. The scrambled query $j \in S_q \setminus C_q$ that achieves the highest score σ_j is added to C_q.
4. For each document i hit by the scrambled query j, update $\beta_i = f \cdot \beta_i$.

The score σ_j as defined in Eq. 4 weighs two criteria for the incremental selection process; the retrieval effectiveness of each potential scrambled query and the current coverage level of each of the documents that have to be covered. A scrambled query with a high precision or high recall is amplified with respect to other candidate scrambled queries. This heuristic achieved the best results in our tests. The overall selection procedure is a greedy set covering algorithm adapted to the needs of our query scrambling approach.

We performed a large set of experiments, which showed that the retrieval results are improved with this guided covering algorithm. The improvement is more evident for larger sample sizes. This is a rather expected outcome, since larger samples can offer more accurate statistics about the underlying collection. We do not present the results of all the above-mentioned experiments here for space reasons; we just present some results with a larger sample size and smaller numbers of retrieved and targeted documents, as an indication of how the methods work for different parameters.

In Tables 5, 6, and 7 we present the comparative results of the MI-based approach (basic) and the covering-based approach when a 20,000 document sample is used. In these experiments, we evaluated the number of target top-10 documents found in the union of the top-100 documents retrieved by all scrambled queries. The results show consistent improvements in almost all cases (italicized) of about 3.1 % on average (3.8 % for ABT, 2.9 % for RG, and 2.3 % for AG), where each improvement is calculated as the average difference between the effectiveness of the covering-based algorithm and the basic algorithm.

The results of the basic method presented in Tables 5, 6, and 7 are generally slightly worse than the results presented in Tables 1, 2, and 3. Since background experiments have shown that increasing the sample size improves effectiveness, we are inclined to blame the worse performance in the latter three tables on the more challenging task of targeting the

[13] This formula has its roots in clustering, where a commonly used rule-of-thumb for the number of clusters v to look for in n data points is $v \approx \sqrt{n/2}$. We apply this rule-of-thumb in reverse: when going for volume v we use the top $n = 2v^2 + 1$ data points (scrambled queries); the $+1$ is immaterial. The problem can indeed also be solved with clustering: first cluster the top-n scrambled queries into v clusters with queries hitting similar sets of documents within each cluster, and then select one representative scrambled query from each cluster.

Table 5 ABT privacy; MI-based (basic) versus covering-based selected scrambled queries

	v	Unfiltered		$k = 1$		$k = 2$		$k = 4$		$k = 8$		$k = 16$	
		aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
Basic	2	.576	.631	.543	.576	.345	.214	.201	.071	.147	.024	.088	.027
	10	.732	.795	.706	.755	.587	.383	.394	.167	.281	.084	.205	.047
	50	.831	.868	.807	.847	.746	.564	.566	.285	.440	.151	.304	.089
Covering	2	*.616*	*.737*	*.573*	*.705*	*.415*	*.240*	*.228*	*.113*	*.195*	*.038*	*.131*	*.032*
	10	*.734*	*.844*	*.714*	*.815*	*.624*	*.427*	*.449*	*.212*	*.345*	*.114*	*.228*	*.062*
	50	.814	*.887*	.799	*.884*	*.767*	*.618*	*.611*	*.353*	*.489*	*.198*	*.323*	*.104*

Table 6 RG privacy; MI-based (basic) versus covering-based selected scrambled queries

	v	$g = g_q$		$g = 2\,g_q$		$g = 4\,g_q$		$g = 8\,g_q$	
		aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
Basic	2	.407	.194	.362	.116	.239	.066	.188	.030
	10	.586	.314	.545	.211	.405	.138	.314	.066
	50	.722	.462	.679	.289	.569	.183	.413	.116
Covering	2	*.438*	*.244*	*.400*	*.173*	*.318*	*.088*	*.203*	*.035*
	10	*.632*	*.365*	*.592*	*.243*	*.459*	*.139*	*.328*	*.070*
	50	*.746*	*.479*	*.708*	*.329*	*.599*	*.189*	*.425*	.101

Table 7 AG privacy; MI-based (basic) versus covering-based selected scrambled queries

	v	$g = .0064$		$g = .0128$		$g = .0256$		$g = .0512$	
		aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
Basic	2	.221	.111	.211	.110	.163	.093	.078	.057
	10	.403	.206	.329	.218	.236	.157	.152	.115
	50	.543	.344	.437	.304	.315	.203	.185	.138
Covering	2	*.295*	.097	*.253*	*.145*	*.191*	*.114*	*.097*	*.094*
	10	*.443*	*.258*	*.371*	*.220*	*.261*	.152	*.167*	*.120*
	50	*.570*	*.347*	*.455*	*.322*	*.331*	*.228*	*.200*	*.158*
	#*q*	80	36	83	51	89	69	94	87

top-10 instead of the top-50, in percentage/fraction terms, in combination with retrieving only the top-100 documents per scrambled query instead of the top-1000. However, we find the setup used in this Section more realistic for web retrieval (where the first page of top-10 results matters most) and it is more efficient (faster).

We would like to note that in the set covering-based approach that we presented, the selection priority of the scrambled queries is a combination of the covering criterion and the statistical MI criterion. Consequently, the concept of set covering is used as an add-on on top of the statistical method. We consider the results of this section as a "proof of concept" that the statistical query scrambling approach can be further enhanced with algorithmic techniques. However, the idea to exploit the combinatorial nature of the

underlying algorithmic problem in order to guide the scrambled query selection process needs further investigation, which is outside of the scope of this work, and may be part of our future work.

7 Evaluating perceived privacy via a user study, and system usefulness

Our privacy criteria are based on statistical considerations of term co-occurrence. In order to investigate the privacy level that the end-users actually perceive, we conducted a user study.

We selected a subset of ten private queries from our experimental setup, reported in Table 8, trying to cover diverse categories from the full 95 query set. For those, we generated scrambled queries using a 20,000 documents sample of ClueWeb09_B. We assumed a web-search setup and targeted the top-10 documents in ClueWeb09_B, retrieving the top-100 per submitted scrambled query. The fraction of the target documents found by the union of results of the best 2 or 10 scrambled queries (according to MI) is shown in the top part of Tables 9 and 10 (row labeled 'Retrieval'), for the two privacy criteria and several privacy levels and methods. The results seem generally better that the corresponding results of Sect. 5, but they are not directly comparable since here we use a larger sample, only 10 private queries, retrieve the top-100 per scrambled query (not the top-1000) and target only the top-10 (not the top-50) results of the remote engine.

Thirty 3rd-year students at the Electrical & Computer Engineering department of Democritus University of Thrace, Greece, participated in the experiment. They were asked to imagine having an information need expressed by each of those queries at a time (together with their categories) and that they want to keep this need private. Then, they were shown a list of scrambled queries and asked to mark those queries that expose their

Table 8 Private queries used in the user study

Query (*category*)	
Acute hepatitis (*medical*)	Symptoms of bone infection (*medical*)
Hacking Yahoo passwords (*crime/cyber-crime*)	How to make bombs (*crime/terrorism*)
Illegal drugs (*addiction*)	Rehabs in Harrisburg PA (*addiction*)
Sex toys (*sex*)	Free porn movies (*sex*)
Wedding invitations (*personal/social*)	Local dating (*personal/social*)

Table 9 ABT retrieval, privacy and usefulness

	v	$k = 1$		$k = 2$		$k = 4$		$k = 8$		$k = 16$	
		aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
Retrieval	2	.800	.790	.340	.320	.230	.160	.110	.000	.030	.000
	10	.960	.950	.800	.540	.480	.260	.290	.170	.220	.010
Privacy	2	.115	.188	.548	.798	.730	.895	.878	.947	.938	.952
	10	.227	.263	.618	.857	.799	.919	.897	.957	.952	.970
Usefulness	2	.201	.304	.420	.457	.350	.271	.196	.000	.058	.000
	10	.367	.412	.697	.663	.600	.405	.438	.289	.357	.020

Table 10 RG retrieval, privacy and usefulness

	v	$g = g_q$		$g = 2\,g_q$		$g = 4\,g_q$		$g = 8\,g_q$	
		aDF	mDF	aDF	mDF	aDF	mDF	aDF	mDF
Retrieval	2	.540	.290	.380	.080	.320	.070	.200	.070
	10	.860	.530	.800	.080	.690	.190	.320	.070
Privacy	2	.348	.837	.658	.950	.713	.975	.885	.987
	10	.514	.866	.685	.975	.808	.986	.916	.992
Usefulness	2	.423	.431	.482	.148	.442	.131	.326	.131
	10	.643	.658	.738	.148	.744	.319	.474	.131

specific need (ABT privacy), and those queries that expose their specific need or expose them in another similar or worse way (RG privacy). The keywords in multi-term scrambled queries were sorted in a decreasing document frequency (i.e. most to least general), trying to achieve a more 'natural' query look (think, e.g., of "bad wolf" vs. "wolf bad").

We did not evaluate AG privacy; in preliminary tests, we found it especially difficult to formulate the right question to the users. AG privacy is not directly connected to the information need but rather to environmental (e.g. societal) and/or cultural factors affecting the user. It seemed that AG privacy could better be used in an interactive environment where users turn the 'knob' until they deem the candidate scrambled queries as 'safe' with respect to their own, possibly completely different, reasons. A user study of AG privacy would have required a completely different setting.

The middle part of Tables 9 and 10 (row labeled 'Privacy') shows the fraction of the submitted scrambled queries that were deemed 'safe' by users. The trade-off between retrieval effectiveness and perceived/achieved privacy is now obvious. At the largest values of the privacy levels shown, we get perceived privacy of 88.5–99.2 %. in this respect, we investigated a good range of parameter values. The numbers also confirm that mDF indeed provides stricter privacy than aDF.

Operationally, the usefulness of such a query scrambling system should be measured by both the privacy and retrieval quality it achieves. Best retrieval with no privacy or full privacy with no target results both defeat the purpose. In this respect, we can provide a measure of usefulness by averaging retrieval and privacy effectiveness; the low part of the tables shows their harmonic average. The harmonic average is biased to the smaller of retrieval and privacy; it is more appropriate than their arithmetic average since it captures better the fact that having one without the other defeats the purpose. Roughly, for ABT, usefulness peaks at around $k = 2$ for both aDF and mDF; but with aDF the system seems usable in a wider k-range from $k = 1$ up to $k = 16$. Similarly for RG, with aDF the system is usable in a wider g-range than with mDF, and usefulness seem to peak between $2g_q$ and $4g_q$.

In any case, the perceived privacy is less than 100 % for many of the private information needs we experimented with, suggesting that such a tool should, at least in some cases, not be run in a fully automatic way. To assure privacy or at least to keep privacy leakage under control at the user side, sets of candidate scrambled queries could be first presented to the user and some of the suggested queries may have to be discarded manually. Such a tool, nevertheless, shows a great potential in pre-selecting good scrambled query sets, minimizing the user effort. Although our privacy criteria are simple and word-frequency based, this study shows that they are strong indicators of what users perceive as private.

7.1 Inter-user agreement

Table 11 shows the total number of scrambled queries evaluated by each user for all ten private information needs and the inter-user agreement on the 'safety' of scrambled queries, per privacy type and method. While the observed agreements are strong (75.3–93.3 %), the expected by-chance agreements are also strong (58.6–91.0 %) revealing that the rating distributions are skewed: the largest percentage of scrambled queries are deemed as private, from the viewpoint of each user. On the one hand, this means that our methods produce mostly 'safe' scrambled queries, especially in the case of RG/mDF (very few scrambled queries are marked as 'unsafe' per user since the expected agreement is 91.0 %) as it was expected by its high privacy seen in Table 10. On the other hand, Fleiss' κ (Fleiss 1971) and average pairwise Cohen's κ (Cohen 1960) show weaker agreements; both these measures correct agreement rates for the rate of chance agreement.

Common guidelines used in the literature for κ values characterize 0.21–0.40 as 'fair' and 0.41-0.60 as 'moderate' agreement. Such guidelines are however by no means universally accepted and they are rather based on personal opinion. Under these common guidelines, we get a moderate agreement for ABT privacy and a fair agreement for RG privacy. In this respect, RG privacy seems like a more 'personal' matter than ABT privacy. Concerning the methods, there is no consistent pattern: mDF produces higher agreement in ABT privacy, while aDF produces higher agreement in RG privacy.

Manning et al. (2008) suggest that "agreement below 0.67 is seen as data providing a dubious basis for an evaluation [of relevance], though the precise cutoffs depend on the purposes for which the data will be used." Inter-judge agreement of relevance has been measured within the TREC evaluations and for medical IR collections: the level of agreement normally falls in the range of 0.67–0.8 and characterized as 'fair'. In any case, all our privacy agreements—as measured with two versions of κ here—fall in the range of 0.228–0.559, thus they are weaker than a typical relevance agreement. Our weaker agreements may be attributed to the notion of privacy being more subjective than relevance.

8 Practical feasibility: a field experiment

In this section, we investigate the applicability of our query scrambling approach on real web search engines. We apply and evaluate query scrambling against the popular web search engine of Google.

In our approach for query scrambling, a prospective user has to build a sample collection of the documents indexed by a search engine, generate scrambled queries for each

Table 11 Inter-user agreement of the thirty users on the 'safety' of scrambled queries, per privacy type

No. of scrambled queries	ABT		RG	
	aDF 463	mDF 482	aDF 365	mDF 371
Observed agreement	.775	.847	.753	.933
Expected agreement	.586	.665	.625	.910
Fleiss' κ	.458	.544	.342	.251
Avg. pairwise Cohen's κ	.468	.559	.341	.228

private query, submit them in a privacy-preserving way to the search engine, and, finally, collect the scrambled results and identify the target results in them. We are well aware that the above steps are not trivial.

One may dismiss this as an unrealistic approach and contend that we cannot expect users to perform all the above steps. However, the protection of user privacy for certain web searches can be of crucial importance for privacy, economic, political, or democratic reasons, and users may take the cost to use query scrambling, if they have the option to do it. Query scrambling is not intended to displace regular web searches; it should only be applied selectively for really sensitive web searches. In this case, the overhead for the user should be well-justified.

In any case, all the steps required can be packaged in an application and performed automatically. Furthermore, document samples of search engines may be centrally distributed, often enough (similarly to signature databases for anti-virus applications), by a provider/vendor who does not have to be necessarily trusted. Updated samples also solve the problem of non-static document collections indexed by search engines. In this sense, our solution is feasible and practical, since a single user with commodity hardware resources can use it; there are no assumptions about special search engines or trusted parties.

One may also doubt the feasibility of the approach when using a real web search engine like Google or Bing and that it will retrieve any useful results. To investigate this issue, we built some prototype components and conducted a field experiment of query scrambling on a real web search engine. We chose Google for the experiment, but other web search engines like Bing or Baidu could also be used.

First, we created two query-based samples of the search engine's document collection. We used the sampling algorithm described in Sect. 3. We built two samples, one of 50K documents and one of about half the size, 23K documents. The small sample is a snapshot taken during the construction of the large one.

To significantly reduce the number of queries submitted to the search engine, a set of 12 private queries was chosen, the number of target results per private query and the number of retrieved results per scrambled query were reduced to 10 and 100, respectively. Finally, we run experiments for scrambled query volume $\upsilon = 50$ and privacy criterion ABT with $k = 1$ using mDF. For this experiment, we chose the minimum privacy in order to investigate the retrieval effectiveness of our approach; additionally, we were more interested in the feasibility of the infrastructure needed and indicative time consumption. The outcome, however, was better than expected, which clearly calls for further experiments with higher privacy guarantees.

The field experiment was executed mainly with the same algorithms and tools that we used in our main experiments in Sect. 5. The first challenge was to submit the scrambled queries in such a way that the queries are unlinkable to each other, i.e., it should not be possible for the search engine to find out which scrambled queries correspond to the same private information need. The second challenge was that we had to retrench our query submissions in order to avoid excessive load on commercial search engines and the risk of our clients being locked out.

There are several ways to deal with these challenges, like for example using the Tor anonymity network, using anonymity proxies, or some crowd-based approach, and none of them was trivial. We opted for a crowd-sourcing approach where a set of volunteers agreed to submit queries on behalf of our query scrambling coordinator. Each volunteer client simply installed a dedicated Chrome extension which took over the interaction with the coordinating server of the experiment.

We managed to build a community of about 30 clients/volunteers, with each client submitting a query every few minutes (while the client was up). The samples were built during November and December 2012, and the private queries were issued in December 2012. The same crowd-sourcing platform was used for both tasks, creating the samples and running the scrambled queries. More precisely, it took about 6 weeks to create both samples, and about a week to run the scrambled queries. The total number of 'transactions' with the search engine for the complete set of query scrambling experiments (not including the generation of the samples) was: (12 private queries) $\times$ (volume 50) $\times$ (10 pages with results for each scrambled query) $\times$ (2 samples) = 12,000, plus the 12 private queries for the ground truth. With an average number of ten clients active at any time and each client submitting one query (requesting ten results) approximately every 8 min, it took less than 7 days to retrieve the results for all scrambled queries from Google. None of the participants reported any significant (or even noticeable) computational load during the experiment. The computational load for a normal user of our query scrambling solution would probably not be different.

We were positively surprised by the high quality of the results, especially when the larger sample was used (Table 12). When using the 50K sample. we retrieved nine or ten target documents for most private queries, and the others do not fall far behind. Admittedly, this is a small-scale experiment with the lightest privacy requirements, but the outcome indicates the feasibility and the potential of our query scrambling approach.

We performed a failure analysis on the two worst-performing queries, i.e. "israel political system" and "how to make bombs". Their inferior performance was traced to the following factors: (a) the documents in our samples, unfortunately, do not seem to cover the topic of "israel political system", pointing to the need of larger samples, and (b) Google is not exactly a bag-of-words engine; phrases like "how to", "what is", and others, seem to have a special meaning and thus weighted more heavily, while in our case these are just bags of high frequency unimportant words which do not influence much the scoring/ranking. Furthermore, while query word order seem to matter in Google, this did not seem to have an adverse effect in our experiment.

Operationally, the long times consumed indicate that document samples should be constructed and distributed to interested first-time or occasional users by a central authority

Table 12 Query Scrambling on Google: number of top-10 target documents found by 50 scrambled queries

Private query	23K sample	50K sample
Acute hepatitis	4	6
Cancer society	4	6
Car radar detectors	7	10
Computer game cracks	5	9
Hacking yahoo passwords	8	7
How can aids be transferred	7	6
How to make bombs	5	5
Illegal drugs	9	9
Israel political system	2	3
Microsoft product keys	9	9
Symptoms of bone infection	6	9
Windows rootkit	1	9
Average	5.6 out of 10	7.3 out of 10

(vendor) or in some community-based manner in regular time intervals, and that many more clients than just 30 should participate in such a crowd-sourcing approach in order to reduce task completion times. Regular users of query scrambling could have their own low priority sampling task constantly running in the background at their side. Then, given the appropriate sample, the scrambling of a single private query could take from a few seconds to several hours, depending on the privacy measures taken for the submission of the scrambled queries.

9 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. More importantly, the proposed approach does not disclose the original search query—and thus the exact search interest of the user—to any party, including the search engine. 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 Arampatzis et al. (2011), 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.

The guided covering algorithm showed that the retrieval effectiveness can be further improved by carefully guiding the scrambled query selection process. Even though the improvements were small, we consider them very important because (a) they occurred in almost all cases we examined, and (b) the size of the sample that was used to guide the covering procedure was minuscule with respect to the size of the collection, nevertheless, covering the sample seemed to still transfer to the whole collection. Furthermore, the field experiment on Google showed that our approach, even though time-consuming, is practicable on real search engines.

While it is easy to evaluate the retrieval effectiveness of our methods, we evaluated the actual privacy perceived by end users via a user study. The study showed that our simple word- and phrase-frequency based criteria can be strong indicators of what users find acceptable and safe from a privacy perspective. A further outcome is that even in cases where the average level of privacy of the scrambling tools is high, it may not be possible to assure 100 % privacy in a fully automatic way. This suggests that users should specify their privacy requirements in a very conservative way, or for even higher assurance, they may opt to review the set of scrambled queries supplied by the system, discarding queries they find ‘unsafe’.

Acknowledgments The material in Sect. 8 was contributed by George Stamatelatos, master's student at the Electrical & Computer Engineering department, Democritus University of Thrace, Greece.

References

Arampatzis, A., Kamps, J., & Robertson, S. (2009). Where to stop reading a ranked list: Threshold optimization using truncated score distributions. In *SIGIR*, *ACM* (pp. 524–531).

Arampatzis, A., Efraimidis, P., & Drosatos, G. (2011). Enhancing deniability against query-logs. In *ECIR*, *Springer*, *lecture notes in computer science* (Vol. 6611, pp. 117–128).

Arampatzis, A., Efraimidis, P. S., & Drosatos, G. (2013). A query scrambler for search privacy on the internet. *Information Retrieval*, *16*(6), 657–679.

Barbaro, M., & Zeller, T. (2006). A face is exposed for AOL searcher no. 4417749. Accessed June 5, 2014. http://www.nytimes.com/2006/08/09/technology/09aol.html.

Bhagat, S., Weinsberg, U., Ioannidis, S., & Taft, N. (2014). Recommending with an agenda: Active learning of private attributes using matrix factorization. In *Proceedings of the 8th ACM conference on recommender systems* (pp. 65–72). New York: ACM. RecSys '14. doi:10.1145/2645710.2645747.

Boneh, D., & Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Theory of cryptography, lecture notes in computer science* (Vol. 4392, pp. 535–554). Berlin: Springer. doi:10.1007/978-3-540-70936-7_29.

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of GSCL* (pp. 31–40). http://www.ling.uni-potsdam.de/ gerlof/docs/npmi-pfd.pdf.

Brown, P. F., Pietra, V. J. D., de Souza, P. V., Lai, J. C., & Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, *18*(4), 467–479.

Callan, J. P., & Connell, M. E. (2001). Query-based sampling of text databases. *ACM Transactions on Information Systems*, *19*(2), 97–130.

Cao, N., Wang, C., Li, M., Ren, K., & Lou, W. (2014). Privacy-preserving multi-keyword ranked search over encrypted cloud data. *Parallel and Distributed Systems, IEEE Transactions on*, *25*(1), 222–233. doi:10.1109/TPDS.2013.45.

Caprara, A., Fischetti, M., & Toth, P. (1998). Algorithms for the set covering problem. *Annals of Operations Research*, *98*, 2000.

Carpineto, C., & Romano, G. (2013). Semantic search log k-anonymization with generalized k-cores of query concept graph. In *Advances in information retrieval, lecture notes in computer Science* (Vol. 7814, pp. 110–121). Berlin: Springer. doi:10.1007/978-3-642-36973-5_10.

Carr, R. D., Doddi, S., Konjevod, G., & Marathe, M. (2000). On the red-blue set cover problem. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete Algorithms* (pp. 345–353). Philadelphia: Society for Industrial and Applied Mathematics. SODA '00, http://dl.acm.org/citation.cfm?id=338219.338271.

Castellà-Roca, J., Viejo, A., & Herrera-Joancomartí, J. (2009). Preserving user's privacy in web search engines. *Computer Communications*, *32*(13–14), 1541–1551. doi:10.1016/j.comcom.2009.05.009.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, *4*(3), 233–235. doi:10.2307/3689577.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37.

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

Domingo-Ferrer, J., Solanas, A., & Castellà-Roca, J. (2009). h(k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, *33*(4), 720–744. doi:10.1108/14684520910985693.

Fleiss, J., et al. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*(5), 378–382.

Hannak, A., Sapiezynski, P., Molavi Kakhki, A., Krishnamurthy, B., Lazer, D., Mislove, A., & Wilson, C. (2013). Measuring personalization of web search. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 527–538). Republic and Canton of Geneva: International World Wide Web Conferences Steering Committee. WWW '13. http://dl.acm.org/citation.cfm?id=2488388.2488435.

Howe, D. C., & Nissenbaum, H. (2009). TrackMeNot: Resisting surveillance in web search. In I. Kerr, C. Lucock, & V. Steeves (Eds.), *Lessons from the Identity trail: Anonymity, privacy, and identity in a networked society* (Chap 23, pp. 417–436). Oxford: Oxford University Press.

Karp, R. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Thatcher (Eds.), *Proceedings of a Symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, The IBM Research Symposia Series, Complexity of computer computations* (pp. 85–103). New York: Plenum Press.

Lindell, Y., & Waisbard, E. (2010). Private web search with malicious adversaries. In *Privacy enhancing technologies, lecture notes in computer science* (Vol. 6205, pp. 220–235). Berlin: Springer. doi:10.1007/978-3-642-14527-8_13.

Lund, C., & Yannakakis, M. (1994). On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, *41*(5), 960–981. doi:10.1145/185675.306789.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press.

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

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: ACM Press.

Peddinti, S. T., & Saxena, N. (2014). Web search query privacy: Evaluating query obfuscation and anonymizing networks. *Journal of Computer Security*, *22*(1), 155–199. http://dl.acm.org/citation.cfm?id=2590636.2590640.

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: ACM.

Sánchez, D., Castellà-Roca, J., & Viejo, A. (2013). Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines. *Information Sciences*, *218*, 17–30. doi:10.1016/j.ins.2012.06.025.

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

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

Terra, E. L., & Clarke, C. L. A. (2003). Frequency estimates for statistical word similarity measures. In *HLT-NAACL, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, May 27–June 1, Edmonton, Canada*.

Tigelaar, A. S., & Hiemstra, D. (2010). Query-based sampling using snippets. In *Eighth workshop on Large-Scale Distributed Systems for information retrieval, Geneva, Switzerland, CEUR-WS, Aachen, Germany, CEUR workshop proceedings* (Vol. 630, pp. 9–14).

Viejo, A., & Sánchez, D. (2014). Profiling social networks to provide useful and privacy-preserving web search. *JASIST*, *65*(12), 2444–2458. doi:10.1002/asi.23144.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *ICML, Morgan Kaufmann* (pp. 412–420).

Young, N. E. (2008). Greedy set-cover algorithms. In M.-Y. Kao (Ed.), *Encyclopedia of algorithms* (pp. 379–381). US: Springer. doi:10.1007/978-0-387-30162-4_175.