Efficient Query Obfuscation with Keyqueries

Maik Fröbe Eric Oliver Schmidt Matthias Hagen
Martin-Luther-Universität Halle-Wittenberg

ABSTRACT

Search engine users who do not want a sensitive query to actually appear in a search engine's query log can use query obfuscation or scrambling techniques to keep their information need private. However, the practical applicability of the state-of-the-art obfuscation technique is rather limited since it compares hundreds of thousands of candidate queries on a local corpus to select the final obfuscated queries. We propose a new approach to query obfuscation combining an efficient enumeration algorithm with so-called keyqueries. Generating only hundreds of candidate queries, our approach is orders of magnitude faster and makes close to real-time obfuscation of sensitive information needs feasible.

Our experiments in TREC scenarios on the ClueWeb corpora show that our approach achieves a retrieval effectiveness comparable to the previous exhaustive candidate generation at a run time of only seconds instead of hours. Overall, 75% of the private information needs can be obfuscated while retrieving at least one relevant document of the original private query—that itself will not appear in the search engine logs. To further improve a user's privacy, the query obfuscation can easily be combined with other client-side tools like TrackMeNot or PEAS fake queries, and TOR routing.

CCS CONCEPTS

• **Information systems** → **Query reformulation**; *Web searching and information discovery*; • **Security and privacy** → **Privacy protections**.

KEYWORDS

Query Obfuscation; Private Information Retrieval; TREC evaluation

ACM Reference Format:
Maik Fröbe, Eric Oliver Schmidt, and Matthias Hagen. 2021. Efficient Query Obfuscation with Keyqueries. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3486622.3493950

1 INTRODUCTION

Privacy and confidentiality are current challenges in the field of information retrieval [12]. One respective research direction addresses the anonymization of query logs [21, 26, 43] to avoid scenarios like the deanonymization of users when the AOL query log was released.[1] However, any such privacy techniques applied on the search engine side require users to trust the search engine. This trust might be unacceptable for users with some very sensitive information need, especially given recent news that the police can access query logs;[2] unthinkable and in conflict with the Bill of Rights if someone used traditional libraries to obtain their information.

User-side query obfuscation techniques can improve privacy and confidentiality without any need to trust the search engine. Corresponding algorithms [4–7] derive alternative queries that are submitted to the search engine instead of the actual query. Hence, the sensitive query itself never appears in the search engine's log, which would happen for other privacy techniques like hiding the actual query in a stream of fake queries—an attacker would then still know that the sensitive query exists. The challenge of query obfuscation is to derive alternative, less sensitive queries that do not directly reveal "secrets" the user perceives as private while still returning results relevant to the original query.

The existing query obfuscation approaches [4–7] derive queries that, on a local corpus, retrieve results similar to the original private query. While those approaches are effective in coming up with good alternative queries, they do so by running hundreds of thousands of candidate queries against the private corpus. To reduce the computational effort and to enable an actual practical applicability with close to real time efficiency, we propose a new approach that generates only hundreds of candidates while maintaining the effectiveness of the previous approaches. Our new approach first retrieves the top results of the private query on the private corpus as target documents and from them extracts promising keywords and keyphrases as the vocabulary for obfuscated queries. To formulate minimal query obfuscation candidates, we employ keyqueries [18, 19] that retrieve (some of) the target documents in the top ranks against the private corpus. Skipping non-minimal queries (in the sense of term sets) and using a practically efficient enumeration algorithm from the field of hypergraph transversal generation, our new approach is orders of magnitude more efficient than the previous obfuscation methods. In a final step, the candidate obfuscation queries are ranked and optionally presented to the user to select the queries that can be submitted to the public search engine. The results of these queries are then locally re-ranked and presented to the user.

We compare our new query obfuscation algorithm to the existing approach of Arampatzis et al. [5] on the ClueWeb09 and ClueWeb12 crawls using 63 TREC Web Track topics with sensitive information needs. In the evaluation, we simulate that users obfuscate queries on a small local corpus and then submit the obfuscated queries to a search engine indexing a disjoint and much larger corpus. We conduct a user study on candidate queries that should be removed to not reveal the private information need. Altogether, ignoring

[1] https://www.nytimes.com/2006/08/09/technology/09aol.html
[2] https://www.cnet.com/news/google-is-giving-data-to-police-based-on-search-keywords-court-docs-show

WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia

ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493950

non-minimal queries, as our efficient enumeration scheme does, comes with a nice regularizing effect: often it even increases the retrieval effectiveness while at the same time reducing the number of generated candidates by 17–19% and dramatically reducing the overall number of locally submitted queries by orders of magnitude. The approach with the best efficiency/effectiveness tradeoff obtains a Precision@10 of 0.38 on par with the state-of-the-art query obfuscation of Arampatzis et al. [5], while on average submitting only 784 candidate queries instead of hundreds of thousands. We make all data and software publicly available to simplify reproducibility.[3]

2 RELATED WORK

We review the existing tools for private web search that improve users' privacy without requiring any trust in search engines, and existing approaches to query obfuscation as an orthogonal technique to current privacy tools.

Tools for Private Web Search. Search engines log user interactions for continuous improvements, e.g., to learn ranking models [23, 28] or to personalize results via user profiles [37]. Collecting the required huge query logs received negative attention [31, 39], and, although not intended, server-side anonymization can fail [24].

Hence, many privacy techniques aim at impeding data collection by search engines, hiding the private query within additional cover queries [3, 14, 31, 32, 39, 42]. TrackMeNot provided the first implementation of such techniques by submitting cover queries sampled from popular queries [31, 39]. Ensuring some semantic similarity allows selecting cover queries similar to the private query on unrelated topics, thus better hiding the private query [30]. However, users may experience suboptimal retrieval performance since the cover queries can introduce noise to their profile [10], which motivates fine-grained adaptions to cover queries following user-defined personalization / privacy tradeoffs [2, 3]. Instead of submitting additional cover queries, PEAS combines the private query with multiple fake queries using a logical OR [33]. Still, those queries might be linked across user sessions, e.g., based on IP addresses [34].

The privacy level introduced by tools like TrackMeNot or PEAS can further be increased in combination with tools for private web browsing. While all major browsers include private browsing modes, they cant provide perfect privacy [1] (e.g., fingerprinting browsers, with features such as the screen resolution, timezone installed fonts etc., is still possible in private browsing modes [1]). Hence, more advanced approaches use anonymous communication via TOR routing [13, 17], advanced strategies to handle cookies [35], or they intercept HTTP communication [34] filtering out identifying information such as cookies or timing attacks. Altogether, those (combinations of) tools improve the privacy of web searchers. However, they still submit the private query to the search engine, revealing the existence of the private information need that ends up as a query in the search engines' log.

Query Obfuscation. Query obfuscation approaches, introduced by Arampatzis et al. [6], replace the private query with a set of less sensitive queries and combine the results at the searcher's end into a single ranking, preventing the private query from being submitted to the search engine. Query obfuscation works in two steps. In the first step, a candidate generation produces candidate queries. Second, a candidate selection chooses the candidate queries submitted to the public search engine. There are semantic and statistical approaches to query obfuscation, using different methods for the candidate generation and selection.

Semantic query obfuscation [6, 7] leverages semantic relations between terms, replacing the private query with sets of queries representing more general concepts. The WordNet taxonomy [29] is used to generate candidate queries by extracting hyponyms, i.e., more general terms with an "is-kind-of" relationship to a query term. Query candidates are selected with a user-defined similarity threshold, where semantic similarity is measured using the distance in WordNet [36], discarding candidate queries above the threshold. Semantic query obfuscation comes with two drawbacks. First, obfuscated queries have a semantic relationship to the private query, introducing the risk of revealing the private information need. Second, the extraction of terms from WordNet does not take into account that some terms are more effective in retrieving relevant documents than others.

Statistical query obfuscation [4, 5] addresses both problems by using a private search engine for the generation and selection of the candidate queries, yielding a better retrieval effectiveness. The private query is submitted to the private search engine, using the top-ranked documents as target documents for candidate generation and candidate selection. During candidate generation, a sliding window covering 16 consecutive terms moves over the target documents, emitting all one-, two-, and three-term combinations occurring in the same sliding window as candidates. Afterwards, the candidate selection removes candidate queries that do not meet the user specified privacy level by removing queries that retrieve too few documents on the private search engine. Finally, the remaining candidate queries with the highest pointwise mutual information for the target documents are submitted to the public search engine.

Arampatzis et al. [6, 7] conducted a recall-oriented evaluation of query obfuscation on the ClueWeb09B corpus using 95 sensitive queries extracted from the AOL query log. In their case, the private search engine indexes 50,000 documents sampled from the ClueWeb09B. Statistical query obfuscation retrieves 56–76% of the top-50 documents retrieved for the private query, compared to 22–25% retrieved by semantic query obfuscation. However, those excellent results come at non-negligible costs: first, the number of candidate queries submitted to the private search engine is huge, and second, Arampatzis et al. apply only a lightweight privacy level that leads to, for instance, obfuscating the private query `gun rack` by the not really less sensitive queries `gun` or `gun light`.

Our new approach to query obfuscation drastically reduces the number of candidates examined locally by statistical query obfuscation while maintaining the good retrieval effectiveness. Additionally, we increase the privacy level (e.g., prohibiting that the query `gun rack` is obfuscated by `gun`) by considering the popular hypothesis that information needs are non-specifiable [8]. We believe that only users can assess if an obfuscated query reveals the private information need or not. Hence, we include a step in which the user must confirm the obfuscated queries before submitting them to the public search engine. We evaluate this setup in a user study, supporting the participants with automatic filtering inspired by semantic query obfuscation.

[3] https://github.com/webis-de/wi21-query-obfuscation-with-keyqueries

3 APPROACH

We first describe exhaustive approaches to generate query candidates (one existing, two new ones) and scoring schemes to select the best candidates (the best from the previous work and a new additional one). To compute the selection scores, all approaches submit all candidates to the local search engine. To reduce that number of local query submissions, we then present our new efficient approach based on the concept of keyqueries and an enumeration algorithm from the field of hypergraph transversal generation.

3.1 Exhaustive Query Candidate Generation

The goal of the candidate generation phase is to identify combinations of terms as query candidates that could be used to replace a given private query. The respective approaches use the top-n results of the private query as the target documents (we set $n = 10$) and try to identify other queries that retrieve many of the targets. We describe the candidate generation approach of Arampatzis et al. [4, 5] and two new candidate generation approaches.

In each candidate generation approach, we suggest to use a filter list to remove terms that could reveal the private information need. The list contains all terms from the private query and, inspired by semantic query obfuscation [6, 7], also all synonyms, hyponyms, and hypernyms from WordNet [29]. Matching possible query terms against the filter list is done by normalizing all terms with Lucene's default tokenization, lower casing, and Porter stemming [40] to also identify inflected forms. By using the filter list, we employ a more restrictive level of privacy than Arampatzis et al. [5], who only ensured that the original private query is not submitted as a whole. Still, the queries `gun` or `gun light` were valid obfuscated queries for the private query `gun rack` in the setup of Arampatzis et al. [5]. Our idea of including a filter list ensures that such revealing candidates are automatically removed. Additionally, our overall obfuscation approach includes a phase where users review the remaining queries to further remove sensitive candidate queries before any query is submitted to the public search engine, ensuring that obfuscated queries do not reveal the private information need.

Sliding Window. Arampatzis et al. [4, 5] proposed to generate candidate queries from terms that appear close to each other in some target document. They move a window of width 16 over a target document's main content (in our case, extracted with boilerpipe [25]). From a window's content, Arampatzis et al. remove all stop words (in our case, Lucene's default stop word list for English). Additionally, we also remove all terms from the filter list since those terms might reveal the private information need. The unique unordered combinations of one, two, or three of the remaining terms are the candidate queries. Arampatzis et al. chose a window width of 16 terms to ensure that the terms are related to each other [38] and they chose combinations of up to three terms to ensure that the number of candidate queries remains feasible.

Still, sliding window candidate generation is a brute force method, with all advantages and disadvantages. On the one hand, the exhaustive nature of the generated candidate queries somewhat gives an empirical upper bound on the retrieval effectiveness. On the other hand, the number of generated candidate queries quickly grows in practice—and all of them are submitted to the private search engine. A single window with 10 unique terms yields $\binom{10}{1} + \binom{10}{2} + \binom{10}{3} = 175$ unordered one-, two-, and three-term combinations. The actual total number of generated queries is not really restricted: the longer the extracted main content of a target document is, the more candidate queries will be created (easily yielding hundreds of thousands of candidate queries as shown in Section 4).

Noun Phrases. To reduce and better control the number of candidates compared to the sliding window approach, our first new candidate generation approach uses the typical components of human keyword queries: noun phrases. From the combined set of target documents, we extract the frequency-wise top-p noun phrases (we set $p = 10$) and generate all combinations from this vocabulary as the candidate queries. In contrast to the sliding window generation, the number of generated candidates now is fixed. A set of p unique noun phrases yields $2^p - 1$ unique non-empty candidate queries (i.e., the power set of the vocabulary without the empty query). All of these candidates are submitted to the private search engine (e.g., 1023 candidates for $p = 10$).

On each target document's main content (extracted by boilerpipe), we apply OpenNLP's part-of-speech tagger[4] to identify groups of two to five consecutive terms that contain only adjectives and at least one noun (i.e., the "noun phrases") because they mimic typical keyword searches. From the resulting noun phrases, we remove the ones that contain a term from the filter list and near-duplicate noun phrases (normalized by lowercasing and stemming; the most frequent original noun phrase from a group of normalized duplicates is kept). From the extracted noun phrases, we choose the ones that are contained in the most target documents (lowercased and stemmed), breaking possible ties by the total number of occurrences in the target documents. This strategy is motivated by the idea that more common phrases tend to be less specific and therefore less sensitive with respect to the private query.

TF-IDF. Our second new candidate generation approach is based on the idea that terms occurring often in a target document but rarely in other documents may help to retrieve the target document at high ranks. Hence, by ordering terms from a target document by their TF-IDF scores (term frequency times inverse document frequency), the most characteristic ones for the target document could be selected as the query vocabulary. To be able to exploit that idea, ideally, the private search engine provides a fast lookup functionality for document and term frequencies. But since standard baseline retrieval systems like BM25 with RM3 [27] work on document vectors, it is also reasonable to assume that the private search engine directly provides access to document vectors.

Given n target documents (we set $n = 10$) and access to document vectors, our new TF-IDF approach selects the t terms from a target document that have the highest TF-IDF score (excluding terms from the filter list; we set $t = 7$). All unique combinations of these terms form the candidate queries for that target document. The maximum number of candidates generated by the TF-IDF approach thus is $n \cdot (2^t - 1)$ (e.g., at most 1270 candidates for $n = 10$ and $t = 7$).

Comparison. Different to Arampatzis et al.'s candidate queries with terms from 16-words windows, the noun phrase and the TF-IDF candidate generation are able to combine terms not occurring close to each other and better control the number of candidates.

[4] https://opennlp.apache.org/

3.2 Scoring Query Candidates

Query obfuscation approaches score candidate queries to select the ones that should be submitted to the public search engine. To this end, the above exhaustive approaches submit each candidate query to the private search engine and then measure some metric for the results. Arampatzis et al. [5] conducted pilot experiments with precision, recall, F-measure, pointwise mutual information (PMI), and normalized PMI. They found that PMI worked best—basically, the number of results that both the private and the candidate query return divided by the product of the individual numbers of results.

In addition to PMI, we also use the normalized discounted cumulative gain (nDCG) [22] where the target documents form the relevant class (information gain of 1, all other documents get 0). The higher the nDCG of some query with respect to the target documents, the higher the target documents are ranked. Also MAP or similar measures could be suited, but we prefer nDCG since MAP nowadays is criticized for a rather unrealistic user model [16].

3.3 Keyquery-Based Candidate Generation

To reduce the number of candidate queries submitted to the private search engine, our new more efficient approach constructs so-called keyqueries [18, 19] for the target documents of the private query from a given vocabulary (e.g., noun phrases or TF-IDF terms). A query q is a *keyquery* for a set of target documents D against a private search engine P, iff q fulfills the following three conditions [19]: (1) every $d \in D$ is in the top-k results returned by P for q, (2) q has at least l results, and (3) no $q' \subset q$ fulfills the first two conditions.

The first two conditions (i.e., the parameters k and l) define the specificity and the generality of a keyquery. We set $k = 10$ and $l = 100$ to ensure that a keyquery does not reveal the private information need by retrieving exactly the target documents. The third condition is a minimality constraint allowing us to skip the submission of non-minimal query candidates to the private search engine. We argue that minimality is important since the private search engine differs from the public search engine in the indexed documents and the retrieval model. The minimality requirement against the private search engine helps to avoid adding terms to a query that is already "good enough" (i.e., that retrieves the target documents at high ranks) and thus helps to mitigate overfitting.

Given some query vocabulary V for the target documents D (e.g., noun phrases or TF-IDF terms), the set $\mathcal{Q} = 2^V \setminus \{\emptyset\}$ represents the possible candidate queries that can be formulated with terms from V. This set $\mathcal{Q}$ might not always contain queries that actually return all target documents in their top-k results (even for large k). Hence, we relax the first keyquery condition and require that a keyquery retrieves at least m of the target documents within the top-k results of the private search engine P (we set $m = 3$ as it worked well in pilot experiments).

Algorithm 1 gives the pseudocode of our keyquery-based candidate generation and scoring. The algorithm efficiently enumerates all queries from $\mathcal{Q}$ while skipping queries that violate the minimality constraint, thus reducing the number of candidates submitted to the private search engine. The algorithm is a modified version of the HBC algorithm [20] (acronym of the authors' names: Hébert, Bretto and Crémilleux) that was originally proposed for the problem of transversal hypergraph generation. Even though not being output-polynomial for that general generation problem [15], the basic algorithmic idea helps us to efficiently enumerate candidate queries up to a specified fixed length of c terms from the vocabulary (we set $c = 3$ for sliding window, $c = 7$ for TF-IDF, and $c = 5$ for noun phrases since this worked best in pilot experiments). A vocabulary extraction approach may produce multiple vocabularies, such as the sliding window scheme creating a vocabulary for each sliding window. In this case, we run the algorithm on each vocabulary and cache results for duplicated query candidates constructed from vocabularies of different windows.

Like the HBC algorithm, our approach works in stages. In a pre-processing stage (lines 5–12), all terms from the vocabulary are identified that already are keyqueries, ensuring that no other query containing such a term will be enumerated (line 12). In the i-th stage (lines 14–27), all valid candidates of length $i + 1$ (i.e., consisting of $i + 1$ vocabulary terms) are generated. By combining only valid candidates from the previous stage (lines 15 and 16), the HBC enumeration scheme ensures that the minimality criterion is not violated. In our case, this reduces the risk that a candidate query reveals the private information need by retrieving too few results (such candidates are pruned in the lines 20 and 23).

Algorithm 1 Keyquery generation using the HBC enumeration scheme

Input: Target documents D
Keyquery parameters k, l, and m
Vocabulary V (sliding window, noun phrases, or TF-IDF)
Query complexity c (length in number of vocabulary terms)
Candidate scoring function *score* (PMI, nDCG)
Output: Keyqueries Q
Priority queue S with *score*-ranked candidate queries

```
1:  Q ← ∅
2:  S ← ∅
3:  C_1 ← V
4:  C_i ← ∅  for i ∈ {2, ..., c}
5:  for all v ∈ V do
6:      R ← results of v against private search engine
7:      if |R| > l then
8:          S.add(v, score(R))
9:          D' ← { top-k results from R }
10:         if |D' ∩ D| ≥ m then        // (i.e., v is keyquery)
11:             Q ← Q ∪ {v}
12:             C_1 ← C_1 \ {v}
13: i ← 1
14: while C_i ≠ ∅ ∧ i < c do
15:     for all q', q'' ∈ C_i with |q' ∩ q''| = i − 1 do
16:         q ← q' ∪ q''
17:         if q has not been generated yet then
18:             if q \ {v} ∈ C_i for all v ∈ q then
19:                 R ← results of q against private search engine
20:                 if |R| > l then
21:                     S.add(q, score(R))
22:                     D' ← { top-k search results from R }
23:                     if |D' ∩ D| ≥ m then        // (i.e., q is keyquery)
24:                         Q ← Q ∪ {q}
25:                     else
26:                         C_{i+1} ← C_{i+1} ∪ {q}
27:     i ← i + 1
```

Table 1: (a) Overview of the private information needs, and (b) the employed search engines for the ClueWeb09 scenario (CW09; private search engine indexes CW12b13, public search engine indexes CW09) and the ClueWeb12 scenario (CW12; private search engine indexes CW09b, public search engine indexes CW12).

(a) Private Information Needs

	Example Topics	
Category	ID	Query
Health (35 topics)	26	lower heart rate
A user wants health advice while	88	forearm pain
hiding a potential disease.	266	symptoms heart attack
Personal (22 topics)	11	gmat prep classes
A user considers a personal change and	18	wedding budget calculator
wants to hide this before it is settled.	57	ct jobs
Law/Crime (3 topics)	61	computer worm
A user wants legal advice while being	62	texas border patrol
able to reject any criminal intent.	214	capital gains tax rate
Family Pets (3 topics)	38	dogs for adoption
A user has problems regarding a pet	50	dog heat
and wants to hide it till a solution.	111	lymphoma in dogs

(b) Search Engines

		Scenario	
		CW09	CW12
Private Engine	Corpus	CW12b13	CW09b
	Documents	52.3 m	50.2 m
	Size	28.9 GB	27.9 GB
	Size (w. vectors)	126.3 GB	126.1 GB
	Retrieval	Anserini [41] in 2 separate variants: BM25 or QLD.	
Public Engine	Corpus	CW09	CW12
	Documents	1.0 b	733.0 m
	Size	3.6 TB	3.1 TB
	Retrieval	ChatNoir [9] with BM25F (multiple fields) and spam removal.	

4 EVALUATION

We experimentally compare the obfuscation methods on TREC Web track topics that possibly are related to sensitive information needs. We conduct a user study to ensure that the obfuscated queries do not reveal the sensitive information need and then submit the obfuscated queries to a public search engine to compare the retrieval effectiveness in Cranfield-style experiments.

4.1 Experimental Design

Our experimentation is based on topics from the TREC Web tracks that used the ClueWeb corpora. We manually reviewed the originally 300 Web track topics for the ClueWeb09 and the ClueWeb12 and find that 63 of them specify information needs that users might want to obfuscate (often health-related). Table 1 (a) gives example topics and the possible categories of private information needs.

Experimental Setup. Our setup represents users with access to a relatively small private search engine who want to obfuscate sensitive information needs against a much larger public search engine. We employ ChatNoir [9] as the public search engine from which users want to hide their private information needs since ChatNoir is a research search engine indexing the ClueWeb09 and ClueWeb12 corpora using main content extraction, language detection, and metadata extraction (keywords, headings, hostnames, etc.). ChatNoir ranks documents by combining BM25 scores of multiple fields (title, keywords, main content, and the full document) and uses SpamRank scores [11] to remove spam. As the private search engine, we use two search engines based on Anserini [41], one with BM25 and one with QLD as the retrieval model (with and without document vectors required for the TF-IDF candidate generation). Anserini employs a document processing very different to ChatNoir (e.g., no main content extraction). Table 1 (b) gives an overview of the search engines used per scenario. For a ClueWeb09 topic, we use ChatNoir's ClueWeb09 index as the public search engine, and Anserini with BM25 or QLD on the ClueWeb12b13 as the private search engines (roles switched for a ClueWeb12 topic). Since the ClueWeb category B subsets only contain the first 50 million English pages from the respective crawl, our setup ensures that the private search engine has a substantially smaller document corpus than the public search engine that also works on a different crawl with a considerably differing retrieval setup. Given the size of less than 30 GB for the indexes of the private search engine, users can operate such a private search engine at home at moderate costs, e.g., on a Raspberry Pi (similar to the PI-hole[5] for ad blocking).

Candidate Generation. We run the three candidate generation approaches with and without using the HBC algorithm on the 63 sensitive topics using a private BM25 or QLD search engine. Table 2 (a) shows the average number of generated candidate queries per topic and the average number of candidate queries that retrieve more than 5 of the $n = 10$ target documents in their top-10 or in their top-100 results (lines 'Recall@... > 0.5'). Unsurprisingly, the sliding window generates huge numbers of candidates, over 250,000 for BM25 and even more for QLD, all of which are submitted to the private search engine for candidate selection. This yields run times of multiple hours even with high parallelization but, not too surprising, also yields the most candidates with a recall above 0.5.

The TF-IDF and noun phrase approaches generate much more feasible numbers of candidates (less than 0.5% of the sliding window while and still some with recall@10 > 0.5). Applying the HBC algorithm further reduces the number of candidates by 17–19%. Table 2 (b) shows the average target document recall@10 and recall@100 scores on the private search engine. The candidate queries on the x-axis are ordered by their per-topic recall (best candidate

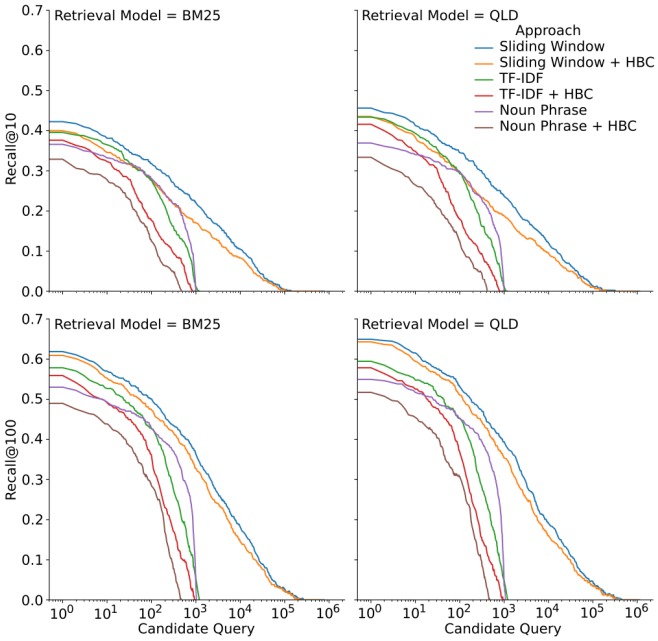
[5] https://pi-hole.net

Table 2: Overview of (a) the number and (b) the estimated quality of the candidate queries generated by the sliding window [5], TF-IDF, and noun phrase approaches without and with HBC. The candidate quality is estimated as the recall of the private query's target documents against the private search engine (either BM25 or QLD as the retrieval model).

(a) Number of Generated Candidates

		Candidate Generation		
		Sliding Window	Noun Phrase	TF-IDF
BM25	Candidates	253,560.1	1,023.0	1,075.8
	(with HBC)	212,409.8	323.9	801.8
	Query length	2.7	12.6	3.6
	(with HBC)	2.6	8.2	3.1
	Recall@10 > 0.5	507.7	67.5	51.8
	(with HBC)	35.4	3.6	8.1
	Recall@100 > 0.5	1,851.6	246.4	144.1
	(with HBC)	979.4	41.8	64.6
	Vocabulary	1,117.7	21.6	41.5
QLD	Candidates	318,088.1	1,023.0	1,044.0
	(with HBC)	266,533.7	316.1	766.8
	Query length	2.7	12.5	3.6
	(with HBC)	2.6	8.0	3.1
	Recall@10 > 0.5	763.5	77.5	57.0
	(with HBC)	87.4	3.0	10.6
	Recall@100 > 0.5	2,719.8	260.8	160.4
	(with HBC)	1,483.5	43.7	74.1
	Vocabulary	1,302.1	21.0	40.5

(b) Estimated Quality of Generated Candidates



of a topic at X=1). The sliding window approach generates candidates with the highest recall—not that surprising for kind of a brute force enumeration—, but the TF-IDF candidates also provide a very good recall given that their number is less than 0.5% of the sliding window candidates. In all cases, generating candidates using QLD achieves slightly better recall, and applying the HBC algorithm reduces the recall—also not surprising since HBC does not expand "already good" queries to avoid overfitting on the private search engine and to have more general obfuscated queries.

User Study. We conduct a user study with computer science students and scholars to identify and remove candidate queries that reveal the private information need. This way, our study resembles an obfuscation setup in which the user confirms any query before it is submitted to the public search engine. A manual inspection is important because information needs are rather non-specifiable [8], resulting in users that may not trust automatic approaches removing sensitive queries for their private information needs.

The study itself consists of two steps. First, for each topic, candidates that contain terms of the private query or synonyms, hyponyms, or hypernyms of a query term are automatically removed to ensure that we invest the annotation budget on candidates that an automatic process cannot detect easily. From the remaining candidates, we collect the top 25 queries of each candidate generation and selection combination that then in the second step are judged by the annotators on whether they might still reveal the private information need. The annotators started by familiarizing themselves with the topic (i.e., reading the query, the description, the narrative, and hints we added on potential privacy threats) and then judging candidate queries in random order.

The annotators assess the candidate queries on a 3-point scale to identify garbage queries (-1), queries that reveal the private information need (0), and queries they would allow to submit to the public search engine (1). The garbage label (-1) indicates low-quality queries, consisting of many spelling mistakes or nonsense words that would make such a query conspicuous for the receiving search engine (e.g., `lickspringscasino` for the topic `lick springs casino`). Contrary to previous studies (cf. Section 2), our annotators were not allowed to consult the public search engine to assess whether a query reveals the private information need since real query obfuscation users would also not want to do so.

During our user study, 21,216 candidate queries were labeled by two annotators. The Fleiss' κ of 0.67 calculated on two topics indicates a good agreement between the annotators. Overall, 252 candidate queries were judged as garbage (label -1), 1,553 as revealing (label 0), and the remaining 19,350 as non-revealing (label 1). Hence, the users in our study would remove 9.3 % of the generated queries (1,805 of 21,216). This emphasizes the importance of manually reviewing the generated candidates. Note that the previous work by Arampatzis et al. [4, 5] neglected this risk by not even automatically removing terms from the private query from queries submitted to the public search engine (e.g., "obfuscating" the query `gun rack` by `gun`). Before our subsequent experiments, we apply the above described automatic filters (synonyms etc.) and remove the candidate queries labeled as garbage or revealing (label 0 or -1) to ensure that the obfuscated queries guarantee high privacy levels.

Table 3: Overview of the average number of retrieved relevant documents ('Relevant') and the Precision@10 ('Prec@10') when 5 (or 20) obfuscated queries are submitted to the public search engine, retrieving the top 10 or top 100 documents from the public search engine per obfuscated query. Results are reported for 3 candidate generation approaches (without and with HBC), nDCG or PMI as candidate selection approaches, and BM25 or QLD as the private search engines' retrieval model.

(a) Top 10 / Top 100 Results for Queries Selected with nDCG

		5 Queries		20 Queries	
		Relevant	Prec@10	Relevant	Prec@10
BM25	Sliding Window	1.31 / 6.39	0.12 / 0.28	3.02 / 10.70	0.21 / 0.34
	+ HBC	1.16 / 6.00	0.11 / 0.29	2.69 / 11.85	0.22 / 0.39
	TF-IDF	0.87 / 3.26	0.09 / 0.21	2.00 / 6.77	0.19 / 0.35
	+ HBC	0.92 / 3.11	0.09 / 0.23	2.11 / 7.69	0.19 / 0.38
	Noun Phrase	0.72 / 1.84	0.07 / 0.12	1.33 / 2.97	0.11 / 0.18
	+ HBC	0.95 / 2.31	0.09 / 0.16	1.64 / 4.02	0.14 / 0.25
QLD	Sliding Window	1.59 / 5.57	0.15 / 0.29	3.31 / 9.39	0.23 / 0.40
	+ HBC	1.18 / 5.49	0.12 / 0.29	3.03 / 10.97	0.22 / 0.40
	TF-IDF	1.18 / 4.28	0.12 / 0.24	2.21 / 7.93	0.21 / 0.37
	+ HBC	1.08 / 4.30	0.11 / 0.25	2.56 / 8.66	0.22 / 0.39
	Noun Phrase	0.74 / 1.75	0.07 / 0.16	1.15 / 2.74	0.12 / 0.18
	+ HBC	0.62 / 2.44	0.06 / 0.17	1.38 / 4.34	0.11 / 0.23

(b) Top 10 / Top 100 Results for Queries Selected with PMI

		5 Queries		20 Queries	
		Relevant	Prec@10	Relevant	Prec@10
BM25	Sliding Window	0.15 / 0.34	0.02 / 0.03	0.52 / 1.79	0.05 / 0.13
	+ HBC	0.20 / 0.69	0.02 / 0.06	1.41 / 5.89	0.13 / 0.29
	TF-IDF	0.84 / 3.52	0.08 / 0.22	1.97 / 7.31	0.16 / 0.34
	+ HBC	0.80 / 3.54	0.08 / 0.23	1.90 / 8.67	0.15 / 0.35
	Noun Phrase	0.89 / 2.49	0.08 / 0.16	1.61 / 3.77	0.15 / 0.23
	+ HBC	1.08 / 3.30	0.10 / 0.20	1.89 / 4.49	0.17 / 0.24
QLD	Sliding Window	0.11 / 0.36	0.01 / 0.03	0.49 / 1.23	0.05 / 0.09
	+ HBC	0.20 / 0.56	0.02 / 0.05	0.93 / 3.33	0.09 / 0.19
	TF-IDF	0.54 / 2.75	0.05 / 0.16	1.62 / 6.23	0.14 / 0.28
	+ HBC	0.49 / 2.56	0.05 / 0.16	1.54 / 6.51	0.13 / 0.30
	Noun Phrase	0.64 / 2.66	0.06 / 0.16	1.26 / 4.11	0.12 / 0.22
	+ HBC	0.62 / 2.70	0.06 / 0.17	1.11 / 4.28	0.10 / 0.22

4.2 Experimental Results

Table 3 shows the retrieval effectiveness of all approaches, submitting 5 or 20 obfuscated queries to the public search engine and retrieving the top-10 or top-100 results per query. All retrieved results are re-ranked on the user-side with Anserini's BM25 retrieval model, measuring the average number of relevant documents retrieved and precision@10 of the final ranking. Selecting candidate queries with the pointwise mutual information achieves lower effectiveness than when using normalized discounted cumulative gain and submitting more obfuscated queries, and retrieving more documents from the public search engine per query improves the effectiveness (the top-100 results for 20 obfuscated queries in Table 3 (a) have the best results). In most cases, the recall observations on the private search engine (Table 2 (b)) transfer to the effectiveness that we observe on the obfuscated queries on the public search engine, i.e., the sliding window approach outperforms TF-IDF, and the noun phrase approach obtains the lowest retrieval effectiveness.

However, there is one noticeable difference: applying the HBC algorithm often improves the effectiveness of obfuscated queries, both in terms of retrieved relevant documents and precision@10. Consequently, the best obfuscation approach using BM25 as the private search engine employs the sliding window approach with the efficient enumeration of candidate queries by the HBC algorithm (retrieving 11.85 relevant documents on average at a precision@10 of 0.39). The sliding window approach is closely followed by the TF-IDF enumeration with HBC (precision@10 of 0.38). Given that this TF-IDF enumeration with HBC achieves almost the same effectiveness as the sliding window approach, it is important to notice that the TF-IDF scheme evaluates only hundreds of candidate queries (Table 2 (a)). In contrast, the sliding window approach investigates hundreds of thousands of candidate queries. Hence, the TD-IDF enumeration with HBC reduces the run time for obfuscating a single query from hours to only a few seconds while achieving similar effectiveness. Using QLD as the retrieval model of the private search engine shows the same trend but yields a slightly better precision@10 of 0.40. Further inspecting the number of topics for which at least one relevant document can be retrieved, we find the maximum of 75 % for TF-IDF and sliding window (both using HBC). This is expected since we apply a rigorous privacy level and some topics only have very few relevant documents making it unrealistic that all information needs can be obfuscated at such a high privacy. Overall, our results show that our new approach, especially combining TF-IDF candidates with the HBC algorithm, achieves state-of-the-art retrieval effectiveness while drastically reducing the number of generated candidates and thus the run time.

5 CONCLUSION AND FUTURE WORK

We have presented an approach inspired by previous query obfuscation techniques. Still, instead of prohibitively low efficiency, the application of an enumeration scheme from the transversal hypergraph field allows for a close to real time query obfuscation since orders of magnitude fewer candidate queries are explored. At the same time, our approach is on a par with respect to retrieval effectiveness with the previous state-of-the-art. While still being kind of "slow search" in the sense that the obfuscation needs a couple of seconds or with user feedback even some minutes, one can argue that in case of a really sensitive information need, users are willing to invest that time—and minutes are really affordable compared to an hour or more the previous state-of-the-art did require.

As interesting directions for future work, one could try to further speed up the process by just taking snippets of the local search into account instead of full documents for the query vocabulary generation. We also plan to implement the approach in an app that can directly be used when wanting to obfuscate queries against any of the big commercial search engines.

REFERENCES

[1] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. 2010. An Analysis of Private Browsing Modes in Modern Browsers. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*. USENIX Association, 79–94.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Intent-Aware Query Obfuscation for Privacy Protection in Personalized Web Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 285–294.

[3] Wasi Uddin Ahmad, Masudur Rahman, and Hongning Wang. 2016. Topic Model Based Privacy Protection in Personalized Web Search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy*, Raffaele Perego, Fabrizio Sebastiani, Javed A. Aslam, Ian Ruthven, and Justin Zobel (Eds.). ACM, 1025–1028.

[4] Avi Arampatzis, George Drosatos, and Pavlos S. Efraimidis. 2013. A Versatile Tool for Privacy-Enhanced Web Search. In *Advances in Information Retrieval - 35th European Conference on IR Research, ECIR 2013, Moscow, Russia (Lecture Notes in Computer Science, Vol. 7814)*, Pavel Serdyukov, Pavel Braslavski, Sergei O. Kuznetsov, Jaap Kamps, Stefan M. Rüger, Eugene Agichtein, Ilya Segalovich, and Emine Yilmaz (Eds.). Springer, 368–379.

[5] Avi Arampatzis, George Drosatos, and Pavlos S. Efraimidis. 2015. Versatile Query Scrambling for Private Web Search. *Inf. Retr. J.* 18, 4 (2015), 331–358.

[6] Avi Arampatzis, Pavlos S. Efraimidis, and George Drosatos. 2011. Enhancing Deniability Against Query-Logs. In *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland*. 117–128.

[7] Avi Arampatzis, Pavlos S. Efraimidis, and George Drosatos. 2013. A Query Scrambler for Search Privacy on the Internet. *Inf. Retr.* 16, 6 (2013), 657–679.

[8] Nicholas J. Belkin. 1980. Anomalous States of Knowledge as a Basis for Information Retrieval. *Canadian Journal of Inf. Science* 5, 1 (1980), 133–143.

[9] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018) (Lecture Notes in Computer Science)*, Leif Azzopardi, Allan Hanbury, Gabriella Pasi, and Benjamin Piwowarski (Eds.). Springer.

[10] Joanna Asia Biega, Rishiraj Saha Roy, and Gerhard Weikum. 2017. Privacy through Solidarity: A User-Utility-Preserving Framework to Counter Profiling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 675–684.

[11] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. 2011. Efficient and Effective Spam Filtering and Re-Ranking for Large Web Datasets. *Inf. Retr.* 14, 5 (2011), 441–465.

[12] J. Shane Culpepper, Fernando Diaz, and Mark D. Smucker. 2018. Research Frontiers in Information Retrieval: Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018). *SIGIR Forum* 52, 1 (2018), 34–90.

[13] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA*, Matt Blaze (Ed.). USENIX, 303–320.

[14] Josep Domingo-Ferrer, Agusti Solanas, and Jordi Castellà-Roca. 2009. H(K)-Private Information Retrieval From Privacy-Uncooperative Queryable Databases. *Online Inf. Rev.* 33, 4 (2009), 720–744.

[15] Khaled M. Elbassioni, Matthias Hagen, and Imran Rauf. 2014. A Lower Bound for the HBC Transversal Hypergraph Generation. *Fundam. Informaticae* 130, 4 (2014), 409–414.

[16] Norbert Fuhr. 2017. Some Common Mistakes in IR Evaluation, and How They Can Be Avoided. *SIGIR Forum* 51, 3 (2017), 32–41.

[17] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. 1999. Onion Routing. *Commun. ACM* 42, 2 (1999), 39–41.

[18] Tim Gollub, Matthias Hagen, Maximilian Michel, and Benno Stein. 2013. From Keywords to Keyqueries: Content Descriptors for the Web. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland*, Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai (Eds.). ACM, 981–984.

[19] Matthias Hagen, Anna Beyer, Tim Gollub, Kristof Komlossy, and Benno Stein. 2016. Supporting Scholarly Search with Keyqueries. In *Advances in Information Retrieval. 38th European Conference on IR Research (ECIR 2016) (Lecture Notes in Computer Science, Vol. 9626)*, Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello (Eds.). Springer, 507–520.

[20] Céline Hébert, Alain Bretto, and Bruno Crémilleux. 2007. A Data Mining Formalization to Improve Hypergraph Minimal Transversal Computation. *Fundam. Informaticae* 80, 4 (2007), 415–433.

[21] Yuan Hong, Xiaoyun He, Jaideep Vaidya, Nabil R. Adam, and Vijayalakshmi Atluri. 2009. Effective Anonymization of Query Logs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China*, David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin (Eds.). ACM, 1465–1468.

[22] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[23] Thorsten Joachims and Filip Radlinski. 2007. Search Engines that Learn from Implicit Feedback. *Computer* 40, 8 (2007), 34–40.

[24] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2008. Vanity Fair: Privacy in Querylog Bundles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA*. 853–862.

[25] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate Detection Using Shallow Text Features. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA*, Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu (Eds.). ACM, 441–450.

[26] Ravi Kumar, Jasmine Novak, Bo Pang, and Andrew Tomkins. 2007. On Anonymizing Query Logs via Token-Based Hashing. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada*. 629–638.

[27] Jimmy Lin. 2018. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52, 2 (2018), 40–51.

[28] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer.

[29] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[30] Mummoorthy Murugesan and Chris Clifton. 2009. Providing Privacy through Plausibly Deniable Search. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, Sparks, Nevada, USA*. 768–779.

[31] Sai Teja Peddinti and Nitesh Saxena. 2010. On the Privacy of Web Search Based Query Obfuscation: A Case Study of TrackMeNot. In *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany*, Mikhail J. Atallah and Nicholas J. Hopper (Eds.), Vol. 6205. Springer, 19–37.

[32] Sai Teja Peddinti and Nitesh Saxena. 2014. Web Search Query Privacy: Evaluating Query Obfuscation and Anonymizing Networks. *J. Comput. Secur.* 22, 1 (2014), 155–199.

[33] Albin Petit, Thomas Cerqueus, Sonia Ben Mokhtar, Lionel Brunie, and Harald Kosch. 2015. PEAS: Private, Efficient and Accurate Web Search. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, Volume 1*. IEEE, 571–580.

[34] Felipe Saint-Jean, Aaron Johnson, Dan Boneh, and Joan Feigenbaum. 2007. Private Web Search. In *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007, Alexandria, VA, USA*. 84–90.

[35] Umesh Shankar and Chris Karlof. 2006. Doppelganger: Better Browser Privacy Without the Bother. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM, 154–167.

[36] Michael Strube and Simone Paolo Ponzetto. 2006. WikiRelate! Computing Semantic Relatedness Using Wikipedia. In *Proceedings of the 21th Conference on Artificial Intelligence, Boston, Massachusetts, USA*. AAAI Press, 1419–1424.

[37] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. 2004. Adaptive Web Search Based on User Profile Constructed Without Any Effort From Users. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA*, Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills (Eds.). ACM, 675–684.

[38] Egidio L. Terra and Charles L. A. Clarke. 2003. Frequency Estimates for Statistical Word Similarity Measures. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada*, Marti A. Hearst and Mari Ostendorf (Eds.). The Association for Computational Linguistics.

[39] Vincent Toubiana, Lakshminarayanan Subramanian, and Helen Nissenbaum. 2011. TrackMeNot: Enhancing the Privacy of Web Search. *CoRR* abs/1109.4677 (2011).

[40] Peter Willett. 2006. The Porter Stemming Algorithm: Then and Now. *Program* 40, 3 (2006), 219–223.

[41] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 1253–1256.

[42] Puxuan Yu, Wasi Uddin Ahmad, and Hongning Wang. 2018. Hide-n-Seek: An Intent-Aware Privacy Protection Plugin for Personalized Web Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 1333–1336.

[43] Sicong Zhang, Grace Hui Yang, and Lisa Singh. 2016. Anonymizing Query Logs by Differential Privacy. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy*, Raffaele Perego, Fabrizio Sebastiani, Javed A. Aslam, Ian Ruthven, and Justin Zobel (Eds.). ACM, 753–756.