The 2016 AAAI Fall Symposium Series:
Privacy and Language Technologies
Technical Report FS-16-04

Safelog: Supporting Web Search and Mining by Differentially-Private Query Logs

Sicong Zhang, Hui Yang, Lisa Singh
Department of Computer Science
Georgetown University
{sz303,gh243,lisa.singh}@georgetown.edu

Li Xiong
Department of Mathematics & CS
Emory University
lxiong@emory.edu

Abstract

Query logs can be very useful for advancing web search and web mining research. Since these web query logs contain private, possibly sensitive data, they need to be effectively anonymized before they can be released for research use. Anonymization of query logs differs from that of structured data since they are generated based on natural language and the vocabulary (domain) is infinite. This unstructured, unbounded data set poses different challenges for producing privacy-preserving query logs. To mitigate these challenges, we propose using a differential privacy framework called Safelog to generate anonymized query logs that contain sufficient contextual information to allow existing web search and web mining algorithms to use the data and attain meaningful results. The key to achieving high privacy guarantees is the introduction of a query pool for augmenting the query log during the sanitation process. We empirically validate the effectiveness of our framework for generating usable, privacy preserving logs for web search, and demonstrate that it is possible to maintain high utility for this task while guaranteeing sufficient levels of privacy. We conclude with a discussion of other web mining tasks that can be supported by these anonymized logs and show some preliminary results for the task of website clustering.

Introduction

In order to advance research in web search and web mining, having access to web search query logs is vital. Large scale web query logs enable researchers to analyze user behavior and improve the quality of web search methodologies. Specifically, web query logs have been used to guide the development of new retrieval methods (Agichtein, Brill, and Dumais 2006; Diriye et al. 2012; Luo, Zhang, and Yang 2014; Zhang, Luo, and Yang 2014), as well as web mining tasks, including website clustering, ranking, and classification, mining semantic relationship, query trend discovery, popular website trend analysis, etc.

However, because query logs are a form of user-generated data, they often contain sensitive and personal information. Releasing query logs without anonymizing them may lead to serious privacy violations. This was the case in 2006 when American Online (AOL) released an anonymized version of their query log (Table 1 shows a sample of the AOL query log) (Adar 2007). In this anonymized log, even though AOL replaced username with user id, the search queries themselves were still very revealing for some users, violating their privacy. The fallout was large (Barbaro and Zeller Aug 2006), and web search companies have hesitated to release any query logs since then, even for research purpose.

A few attempts have been made to alleviate the lack of available search log data. For example, in 2014 Yandex shared an anonymized query log (Table 2) for a web search challenge at the Web Search Click Data (WSCD) 2014 workshop to support document reranking.[1] In this released query log, all words were converted to hash codes, reducing the utility of the released log significantly. The only web search task that can be researched with this data set is document re-ranking (Cai, Liang, and de Rijke 2014; Radlinski and Joachims 2005; Shokouhi et al. 2013). Because the contextual data has been removed, the data set is not useful for studying other traditional web mining tasks, including clustering, classification, and trend analysis.

While not obvious, query log anonymization is more difficult than other more structured data sets because query logs are generated from billions of individual users' natural language. The associated vocabulary domain for these queries is, therefore, infinite. This is a sharp contrast to the finite domains of more traditional data, e.g. itemset mining of a finite domain of items (Lee and Clifton 2014; Cheng et al. 2015).

In this paper, we develop a novel ϵ-differential privacy framework called Safelog that sanitizes and anonymizes a query log. The generated query log maintains utility for web search and web mining algorithms while maintaining strong privacy guarantees. The key to achieving the strong privacy guarantees is the introduction of a *query pool* for augmenting the query log during the anonymization process. We explain and empirically show the privacy guarantee and how to measure the actual retrieval utility for the task of web search (the primary task that uses query logs). We also consider other web mining tasks that can be supported by these anonymized logs and show some preliminary results for a clustering task.

[1] http://research.microsoft.com/en-us/um/people/nickcr/wscd2014/

Table 1: The query log from AOL (sample).

UserID	Query	Query Time	Rank	Clicked Web Page
479	family guy movie references	03-03 22:37:46	1	http://www.familyguyfiles.com
479	top grossing movies of all time	03-03 22:42:42	1	http://movieweb.com
479	top grossing movies of all time	03-03 22:42:42	2	http://www.imdb.com
479	car decals	03-03 23:20:12	4	http://www.decaljunky.com
479	car decals	03-03 23:20:12	1	http://www.modernimage.net
...	...	...	...	...

Table 2: The query log from WSCD 2014 (sample).

SessionID	SERPID	QueryID	ListOfURLs
34573630	0	10509813	34175267 34171511 35444452 15370141 31342884 43630531 26065978 29902424 39016998 62861215
34573635	0	8447254	44298735 41815016 62677540 13753389 3336907 67724115 22354391 4606079 37985498 53161116
...	...	...	...

To summarize, the main contributions of the paper are as follows: (1) This paper is the first to evaluate the utility of differentially private anonymized query logs on the task of web search. We present Safelog, an effective framework for implementing and evaluating both privacy and utility of an anonymized query log. To better evaluate the effectiveness of the anonymized query log, we propose a new utility function that is tailored to this task. (2) We demonstrate how our log anonymization algorithm achieves ϵ-differential privacy, improving the state-of-the-art in this area from (ϵ, δ)-differential privacy (Korolova et al. 2009). (3) We present an empirical analysis that highlights the effectiveness of our framework for document retrieval on real world data. We also analyze the privacy-utility tradeoff so that companies can decide on the level of privacy that is acceptable to them. Based on both the theoretical and empirical findings, we make practical recommendations for companies interested in releasing anonymized query logs that include a detailed discussion of how to set parameters. (4) We discuss other web mining tasks that can be supported by these anonymized logs and show some preliminary results for the task of website clustering.

Related Work

The query log anonymization task came on researchers' radar in 2006 when a user was identified from the released AOL search log (Barbaro and Zeller Aug 2006). For years, researchers have proposed many ad-hoc techniques to help preserve privacy in query logs. (Adar 2007) and (Gotz et al. 2012) proposed anonymizing query logs by removing unique queries and segmenting the search sessions. Jones et al. (Jones et al. 2007; 2008) studied the application of simple classifiers for identifying gender, age, and location, which can largely reduce the size of user candidates for portions of the query log. They found that the re-identification approach remains very accurate even after removing unique terms from the query log. These works verified the need for more robust anonymization techniques for query log anonymization.

Though with limitations, k-anonymity (Carpineto and Romano 2013; Adar 2007; Hong et al. 2009) provides specific privacy guarantees and has been utilized to help in this query log anonymization task. (Carpineto and Romano 2013) and (Hong et al. 2009) proposed methodology to reduce the large-scale data losses and utility. However, the privacy of k-anonymity is based on assumptions about the background knowledge of the adversary. An approach that does not require such strict assumption would be preferred.

Differential privacy (Dwork 2008; Fan et al. 2014; Feild, Allan, and Glatt 2011; Korolova et al. 2009; Gotz et al. 2012) is a promising option since it does not make assumptions about the adversary. Although none of the previous research has achieved a proven private query log anonymization scheme that can publish the query log in its original plain format, some have begun investigating differential privacy in this context. For instance, (Korolova et al. 2009) proposed an algorithm that releases a query-click graph containing queries, clicked URLs with each query, and the corresponding counts. They gave an (ϵ, δ)-differential privacy approach which maintains some utility. However, as mentioned in their paper as limitations, their framework can not output queries that were not included in the original query log, which also means that they cannot achieve ϵ-differential privacy. In this paper, we filled this important gap by 1) proposing a method that preserves more contextual information than previous methods, 2) proposing a utility function that is specific to the primary task of query logs (web search) and leads to a more comprehensive evaluation, and 3) achieving ϵ-differential privacy by incorporating the idea of an external query pool. We accomplish this by using an external query pool.

A closely related research topic to query log anonymization is publishing transaction or sequence data with differential privacy (Lee and Clifton 2014; Cheng et al. 2015). Query log can be viewed as a set-valued or sequence dataset where each user's record corresponds to a set or sequence of query terms and URLs (items), and the goal is to publish the count of query terms or sequences (itemsets). While many algorithms have been proposed in the literature for frequent itemset mining and frequent sequence mining with DP, a fundamental difference is that these methods assume a finite domain for the items, begin with all items in the domain as candidate itemsets and compute their noisy count. While in query logs, there is an infinite possible set of query terms. Hence, the previous work only achieves the weaker epsilon-delta DP by releasing the noisy count of a subset query terms from the query log (as opposed to the entire domain of possible query terms).

Background & Preliminaries

This section gives background and preliminaries to our work. Firstly, a query log Q contains data about requests the search engine receives from its users. The types of data it contains include user ids, user queries, a ranked list of document ids (or URLs to those documents) that the search engine returns to the user, snippets of those documents, the URLs that are clicked by the users, and the timestamps for all the user actions. Then, given an input query log Q, the query log anonymization process is to produce an anonymized query log Q' that has clearly specified privacy guarantees.

Algorithm 1 Query Log Anonymization Algorithm

```
**Input.** $Q$: Original query log collected by search engine; $Q_p$: Set of external search queries; $\tau$ :query filtering parameter; $q_f, c_f$: limiting activity parameters for number of queries per user and number of clicked urls per user; $b, b_q, b_c$: noise parameters; $K$: threshold of tail.
**Output.** $Q'$.
$Q_{clean}$ = removeSensitiveData($Q, \tau$)
$Q_{clean}$ = limitUserActivity($Q_{clean}$,$q_f, c_f$)
$Q^+ = Q_{clean} + Q_p$, considering queries from $Q_p$ as queries with 0 frequency as in $Q_{clean}$.
$Q_{reduced}$ = selectFinalQuerySet($Q^+$,$b, K$)
$Q'$ = generateLogStats($Q_{reduced}$,$b_q, b_c, K$)
$Q'$ = generateQueryTransitions($Q'$,$b_t$)
return $Q'$
```

In order to evaluate the quality of Q' to support web search and mining tasks, we define the utility function $U(Q')$. The function quantifies the ability of Q' to return a list of documents or URLs D that are relevant to for queries $q \in Q'$. Here the utility function corresponds to the evaluation metrics of the document retrieval task.

Most importantly, we use differential privacy in our work. Differential privacy is usually used in an interactive setting in which a statistical database is repeatedly queried. Our problem setting differs from this since our goal is to release a query log. Therefore, we fit into a data publishing or non-interactive setting. When using differential privacy ideas in this context, we need to ensure that an adversary that has a copy of the anonymized query log Q' cannot determine whether or not a particular user exists in the original data set Q. Assume that are two neighboring query logs that only differs by one user's search records. We say that an anonymization scheme $A(Q)$ is (ϵ, δ)-*Differentially Private* if the following definition holds.

Definition. A non-interactive query log anonymization mechanism $A(Q)$ satisfies (ϵ, δ)-*differential privacy* if for all neighboring query logs Q_1 and Q_2, and for all possible output Q^*, the following inequality holds:

$$Pr[A(Q_1) = Q^*] \leq e^{\epsilon} \times Pr[A(Q_2) = Q^*] + \delta \qquad (1)$$

where Q_1 and Q_2 are defined as neighboring when they exactly differ in one user's search log. The goal of a successful data anonymization plan is to have $|U(Q) - U(Q')| < \sigma$, where σ is a small non-negative number; while at the same time ensuring that the privacy level $\epsilon : Q' = A(Q)$ is small enough to maintain high privacy.

The Query Log Anonymization Algorithm

Algorithm 1 shows our high-level algorithm for query log anonymization. The remainder of this subsection describes the main components of the proposed algorithm.

In Safelog, we first empirically remove all queries with a frequency less than 5 from the corpus in order to prevent the release of unique sensitive data. This step also removes typos. We refer to the output of this step as Q_{clean}.

Then, we reduce each user's sample in the query log by limiting the number of queries and URL clicks of each user. Specifically, we only keep the first q_f queries and the first c_f URL clicks of each user from Q, and remove the rest. Intuitively, this step allows us to guarantee that the removal or addition of a single individual in Q has a limited effect on the query log. We will give an experiment later about the values of q_f and c_f.

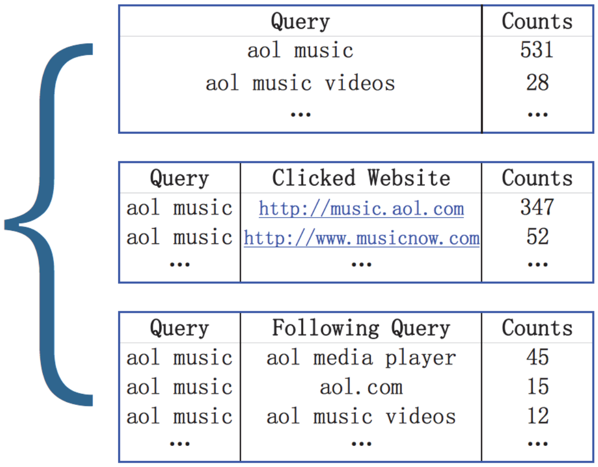
Next, in order to overcome the challenge of the infinite domain in query logs, our key idea is to use an external stochastic query pool to augment the query terms already in the query log. In other words, our query term domain can be viewed as a sampled set of terms S from the set of all possible query terms in the population P. We will show formally that using an external stochastic query pool Q_p to augment the query log with additional queries, improves the overall privacy and allows us to achieve pure DP. We refer to the expanded set of queries as Q^+, where $Q^+ = Q_{clean} + Q_p$. In the next section, we will discuss different query pool generation strategies.

After that, we select the final set of queries to release. Using $Lap(b)$ to represent a random real value drawn independently from the Laplace distribution with mean 0 and scale parameter b (Dwork et al. 2006), we define perturbed counts to be query counts after applying Laplacian noise. We choose to release a query q when its perturbed query count $(M(q, Q^+) + Lap(b))$ is greater than a threshold K, where $M(q, Q^+)$ is the frequency of query q in Q^+. Specifically, for each query q added from the query pool, $M(q, Q^+) = 0$. However, its perturbed query count $(M(q, Q^+) + Lap(b))$ still get a chance to pass the threshold K and therefore be included in the output of this step. Theoretically, since every query on Q^+ has a chance of being selected in the final log, we can achieve ϵ-differential privacy. The final query set generated after this step is referred to as $Q_{reduced}$.

As previously mentioned, we release the perturbed query counts $(M(q, Q+) + Lap(b_q))$ for each query. It is worth noting that for the perturbed query counts, we add noise again using another parameter b_q. Although b_q does not necessarily differ from b, this re-fuzz process reduces the impact of the cut-off threshold K from the previous step. We also release the perturbed click counts for each URL: <q, u, #u was clicked when q was posted + $Lap(b_c)$>. These statistics are the basis for the first two components of the released query log as shown in Figure 1.

Furthermore, releasing query transition information allows us to preserve some sequential information from the original log. It improves the utility of the query log by allowing for more complex web search research, e.g. session search. In our approach, we release adjacent query transitions from Q_{clean} with perturbed counts with a noise scale of $Lap(b_t)$, for each query pairs in $Q_{reduced}$. Specifically, if either of the two queries comes from Q_p, such perturbed counts of query transition will be $0 + Lap(b_t)$ since there are no such transitions in the original log.

Figure 1 shows an example of our proposed format for Q'. Each released Q' consists of three components. The first component contains the released search queries and their corresponding frequencies in Q. The second component contains click-through data for each of the released query-URL pairs. In this part, each line shows a query; a



Query	Counts
aol music	531
aol music videos	28
...	...

Query	Clicked Website	Counts
aol music	http://music.aol.com	347
aol music	http://www.musicnow.com	52
...	...	...

Query	Following Query	Counts
aol music	aol media player	45
aol music	aol.com	15
aol music	aol music videos	12
...	...	...

Figure 1: Format of our anonymized query log with 3 components.

clicked-through URL with this query and the number of clicks for the query-URL pair. The last part of Q' contains information about the query transitions in Q. Specifically, each line shows a pair of adjacent queries along with the frequency of the specific query transition.

Query Pool Generation

Formally, we define Q_p as an external query pool generated using an external set of search queries that are independent of the queries in the original query log Q. Q_p serves as a proxy for the full set of queries that exist in the population P. Each query in Q_p has an equal probability of being included in the query pool. When a commercial search engine using our algorithm, this query pool Q_p can be generated using a random sample of all their recorded queries, or by using queries from a different period. If the previous set of recorded queries is insufficient to represent P, query terms can be randomly extracted from a random set of web pages. Then, we can expand the query log as $Q^+ = Q_{clean} + Q_p$. Queries added to Q^+ from Q_p, are queries with a click count of zero in the original query log.

Because academic researchers do not have access to an extensive query set like commercial companies, we must have an approach for simulating the query pool construction process. Therefore, we propose a simulation algorithm that generates a query pool using artificial queries constructed by randomly sampling and combining high-frequency n-grams present in the English language. In our experiments, we use the Corpus of Contemporary American English (COCA)[2], which includes approximately 450 million words and 190,000 texts. Using this corpus, Davies (Davies 2011) published the (approximately) 1 million most frequent n-grams each for n=2, 3, 4 and 5. We identified 1,159,938 n-grams from this list that end with a noun since nouns are more likely to be part of search engine queries. We sample these n-grams to generate the final query pool Q_p. In other words, we combine the query terms from two independent samples, making it difficult for an adversary to know clearly which queries are real and which ones are not. Using a query pool to maintain log privacy is one of the main contributions of this paper. We will show in our empirical evaluation that even with the addition of these noisy, external data, we can still maintain reasonable utility for web search queries.

Proof of Differential Privacy

We now present a general privacy proof sketch that analyzes the privacy guarantees of our approach for all the major steps in Algorithm 1, except the query transition generation step. Let K, q_f, c_f, b, b_q, b_c and b_t be parameters in our algorithm as defined previously. Let Q be the original query log as input to the algorithm, Q_{clean} be the set of queries from Q that are possible options for release because they occur often enough while keeping at most q_f queries and c_f clicks from each user. Let Q_p be an externally generated stochastic query pool containing a large set of queries. Suppose each possible query q in the infinite domain has a probability of $p_g \in$ [0,1] to be included in the pool Q_p. In practice, the value of p_g depends on the source that is used to generate the query pool. While we provide an approach for generating Q_p, major commercial search engines that have access to a large number of historic queries in their system can create a large pool Q_p satisfying a p_g value close to 1. Here we state the following theorem:

Theorem 1: The query log anonymization algorithm presented in Algorithm 1 satisfies ϵ-differential privacy, where ϵ is defined as:

$$\begin{aligned} \alpha &= Max\{\frac{e^{1/b}}{p_g}, 1 + \frac{1}{2e^{(K-1)/b} - 1}\} \\ \epsilon &= q_f \cdot ln(\alpha) + q_f/b_q + c_f/b_c + (q_f - 1)/b_t \end{aligned} \tag{2}$$

In order to prove Theorem 1, we first consider the following theorem:

Theorem 2: The generation of $q_{reduced}$ in Algorithm 1 satisfies $(q_f \cdot ln(\alpha))$-differential privacy.

Theorem 2 is necessary for our algorithm to achieve ϵ-differential privacy rather than (ϵ,δ)-differential privacy. It makes our algorithm different from previous work (Korolova et al. 2009), and helps us achieve stronger privacy guarantees. Being more specific, our Theorem 2 achieves stronger privacy guarantees than the *Select-Queries* procedure in (Korolova et al. 2009), while the remainder of our algorithm has a similar structure to theirs in terms of adding Laplacian noises. Theorem 1 is now a straightforward proof if we combine our Theorem 2, and Lemmas 2,3, and 4 as presented in (Korolova et al. 2009).

Recall that Q' also contains query transitions. While each user may have at most q_f queries, the user may bring at most $q_f - 1$ adjacent query transitions. Therefore, we can calculate the privacy guarantees in a similar way as the other components. We now prove Theorem 2, thereby showing why we achieve such a stronger privacy notion.

[2] http://corpus.byu.edu/coca/

Proof of Theorem 2

Proof. 1) We first consider the case in which q_f =1, K≥1. Q_1, Q_2 are two neighboring search logs, Q_2 has one more user than Q_1, since $q_f = 1$. This means that Q_2 has one more query q^*. Also, we partition any set of query sets $\hat{Q}$ into two subsets: $\hat{Q}^+$, the query sets in $\hat{Q}$ that contains q^*, and $\hat{Q}^-$, the query sets in $\hat{Q}$ that do not contain q^*. The proof structure is similar to the Lemma 5 proof in Korolova et al. (Korolova et al. 2009). Here we only show why our algorithm has a stronger privacy guarantee than theirs. When $q^* \in Q_1$, we can prove that the algorithm satisfies $(1/b, 0)$-differential privacy using a similar idea as in Korolova et al. (Korolova et al. 2009). Here we give the derivatives in the case of $q^* \notin Q_1, q^* \in Q_2$. For all $q^* \in Q_p, q^* \notin Q$, we have $M(q, Q) = 0$, which is different from Korolova et al. (Korolova et al. 2009). Differential privacy requires the following two inequalities.

$$P[A(Q_1) \in \hat{Q}] \le \alpha P[A(Q_2) \in \hat{Q}] + \delta \tag{3}$$

$$P[A(Q_2) \in \hat{Q}] \le \alpha P[A(Q_1) \in \hat{Q}] + \delta \tag{4}$$

i). For inequality 3: First, we consider the case when q^* is not included in the output. Then $P[A(Q_2) \in \hat{Q}^-]$ = P[q^* not released by $A(Q_2)$] $\cdot$ $P[A(Q_1) \in \hat{Q}^-]$. Therefore,

$$\frac{P[A(Q_1) \in \hat{Q}^-]}{P[A(Q_2) \in \hat{Q}^-]} = \frac{1}{P[q^* \notin A(Q_2)]} = \frac{1}{1 - 0.5exp(\frac{1-K}{b})} \tag{5}$$

Next, we consider the other case when q^* is included in the output. Since $q^* \notin Q_1$, $A(Q_1) \in \hat{Q}^+$ will only be possible when q^* is outputted from the query pool Q_p. Therefore,

$$\begin{aligned} &P[A(Q_1) \in \hat{Q}^+] \\ &= P[q^* \in Q_p] \cdot P[0 + Lap(b) \ge K] \cdot P[A(Q_1) \in (\hat{Q}^+ \setminus q^*)] \\ &= p_g \cdot 0.5exp(-\frac{K}{b}) \cdot P[A(Q_1) \in (\hat{Q}^+ \setminus q^*)] \end{aligned} \tag{6}$$

On the other hand, $q^* \in Q_2$, which means $A(Q_2) \in \hat{Q}^+$ requires q^* be outputted from Q_2:

$$\begin{aligned} &P[A(Q_2) \in \hat{Q}^+] \\ &= P[1 + Lap(b) \ge K] \cdot P[A(Q_1) \in (\hat{Q}^+ \setminus q^*)] \\ &= 0.5exp(\frac{1-K}{b}) \cdot P[A(Q_1) \in (\hat{Q}^+ \setminus q^*)] \end{aligned} \tag{7}$$

Therefore, we achieve an upper bound that:

$$\begin{aligned} &\frac{P[A(Q_1) \in \hat{Q}]}{P[A(Q_2) \in \hat{Q}]} = \frac{P[A(Q_1) \in \hat{Q}^+] + P[A(Q_1) \in \hat{Q}^-]}{P[A(Q_2) \in \hat{Q}^+] + P[A(Q_2) \in \hat{Q}^-]} \\ &\le Max\{\frac{P[A(Q_1) \in \hat{Q}^+]}{P[A(Q_2) \in \hat{Q}^+]}, \frac{P[A(Q_1) \in \hat{Q}^-]}{P[A(Q_2) \in \hat{Q}^-]}\} \\ &= Max\{p_g \cdot \frac{exp(-\frac{K}{b})}{exp\frac{1-K}{b}}, \frac{1}{1 - 0.5exp(\frac{1-K}{b})}\} \\ &= \frac{1}{1 - 0.5exp(\frac{1-K}{b})} \end{aligned} \tag{8}$$

The last step is because $p_g \in [0, 1)$, $exp(-\frac{1}{b}) \in (0, 1)$. Hence we get $p_g \cdot exp(-\frac{1}{b}) < 1 < \frac{1}{1-0.5exp(\frac{1-K}{b})}$. Therefore, inequality 3 holds for $\alpha = \frac{1}{1-0.5exp(\frac{1-K}{b})}$, $\delta = 0$

ii). For inequality 4:

Here we give a stronger upper bound in this case with the help of the query pool Q_p.

$$\begin{aligned} &\frac{P[A(Q_2) \in \hat{Q}]}{P[A(Q_1) \in \hat{Q}]} = \frac{P[A(Q_2) \in \hat{Q}^+] + P[A(Q_2) \in \hat{Q}^-]}{P[A(Q_1) \in \hat{Q}^+] + P[A(Q_1) \in \hat{Q}^-]} \\ &= \{0.5exp(\frac{1-K}{b}) \cdot P[A(Q_2) \in \{\hat{Q}^+ \setminus q^*\}] \\ &+ (1 - 0.5exp(\frac{1-K}{b})) \cdot P[A(Q_2) \in \hat{Q}^-]\}/\{0.5p_g \cdot \\ &exp(-\frac{K}{b}) \cdot P[A(Q_1) \in \{\hat{Q}^+ \setminus q^*\}] + P[A(Q_1) \in \hat{Q}^-]\} \\ &\le Max\{\frac{0.5exp(\frac{1-K}{b})}{0.5p_g \cdot exp(-\frac{K}{b})}, 1 - 0.5exp(\frac{1-K}{b})\} \\ &= Max\{\frac{exp(1/b)}{p_g}, 1 - 0.5exp(\frac{1-K}{b})\} = \frac{exp(1/b)}{p_g} \end{aligned} \tag{9}$$

Therefore, inequality 4 holds for $\alpha = \frac{e^{1/b}}{p_g}$, $\delta = 0$.

Combining the 2 cases, we conclude that our algorithm satisfies the $(ln(\alpha), 0)$-differential privacy, where:

$$\begin{aligned} \alpha &= Max\{e^{1/b}, \frac{1}{1 - 0.5exp(\frac{1-K}{b})}, \frac{e^{1/b}}{p_g}\} \\ &= Max\{\frac{e^{1/b}}{p_g}, 1 + \frac{1}{2exp(\frac{K-1}{b}) - 1}\} \end{aligned} \tag{10}$$

which concludes the proof when $q_f = 1$.

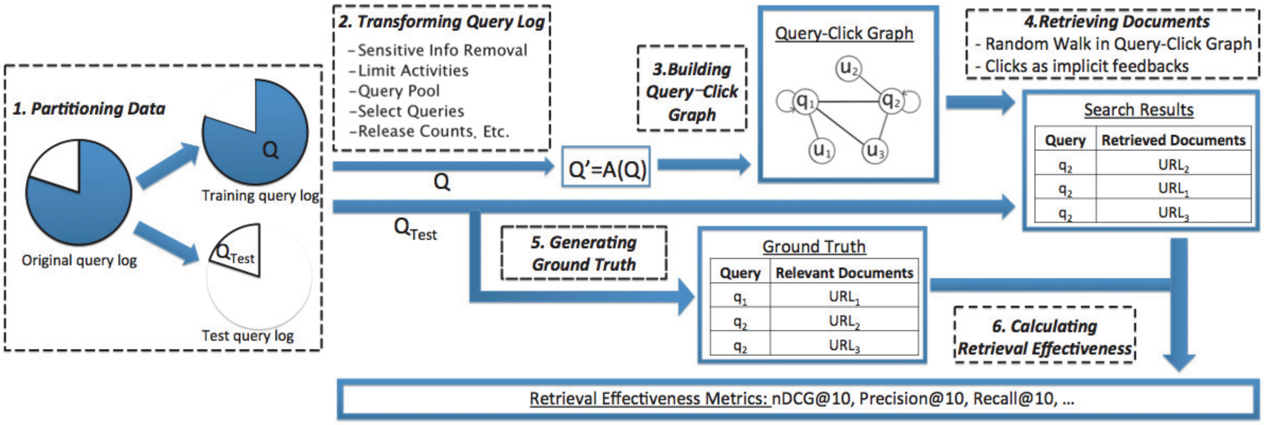
2) Now we generalize the proof for cases when $q_f > 1$, which leads the approach from record level differential privacy to user level differential privacy. While our algorithm achieves differential privacy with $\delta = 0$, the generalization to situations with arbitrary q_f values becomes pretty straightforward. Since Q_1 and Q_2 differs by one user, without loss of generality, suppose Q_2 contains one more user, i.e. it contains q_f additional queries at most, namely q_1, q_2, ...q_{q_f}. Then we have the following:

$$\begin{aligned} P[A(Q_1) \in \hat{Q}] &\le \alpha \cdot P[A(Q_1 + q_1) \in \hat{Q}] \\ &\le ... \le \alpha^{q_f} \cdot P[A(Q_2) \in \hat{Q}] \end{aligned} \tag{11}$$

This concludes the proof of Theorem 2 that the generation of $Q_{reduced}$ is user level $(q_f \cdot ln(\alpha))$-differentially private.

Web Search Task and Utility Function

Figure 2 shows the Safelog framework. It focuses on a workflow of creating anonymized query logs as well as measuring the logs' utility in a complete pipeline for the task of web search. We first partition data using 5-fold cross validation. In each run, we use 80% of the data as the training set Q and the remaining as the test set Q_{Test}. Q acts as the raw query log from the search engine that is the input to the query log anonymization algorithm. Q_{Test} is used to evaluate the utility of our approach on the document retrieval task. Then we



Query	Retrieved Documents
q_2	URL_2
q_2	URL_1
q_2	URL_3

Query	Relevant Documents
q_1	URL_1
q_2	URL_2
q_2	URL_3

Figure 2: Framework Overview.

transform the query log Q into $Q' = A(Q)$ in a privacy-preserving way. After that we build a query-click graph, where nodes are queries and URLs (documents) while edges connect query nodes with their clicked URL nodes.

Retrieving Documents. We retrieve documents using three different algorithms (Agichtein, Brill, and Dumais 2006; Craswell and Szummer 2007; Tran et al. 2015) for queries in Q_{Test} in order to evaluate the utility of the released query log Q'.

The first retrieval algorithm is based on a random walk on the query-click graph. In the graph, nodes are queries and documents (URLs), while the transition weights between nodes are defined by their relationship in the released data. The most common transition type links a query node to a document node. Another type of transition we consider is between two query nodes in the query-click graph. Each query node also has a transition weight to itself as a self-loop. We calculate the transition probability $P(k|j)$ from a query node j to document node or another query node k in a slightly different way from a popular random walk click model proposed by Craswell and Szummer (Craswell and Szummer 2007). Then, we rank the URLs according to the descending order of the probabilities of staying at corresponding URL nodes. $P(k|j)$ is calculated as $(1-s)C_{jk}/\Sigma_i C_{ji}$ when $k \neq j$ and is s when $k = j$. Here C_{jk} is the weight between nodes j and k given by Q', and s is the self-transition probability. If both nodes j and k are query nodes, weight C_{jk} is defined as the query transition counts from j to k as specified in Q'; otherwise, if j is a query node while k is a document node, weight C_{jk} is defined as the click-through counts for this query-document pair as specified in Q'. In our approach, we empirically set the self-loop probability $s = 0.1$.

The second algorithm uses the impact factor of each web page to enhance the random walk model. It is based on (Tran et al. 2015). In this setting, more popular websites gain greater probabilities for the random walker to walk into. An impact factor F for each web page is introduced to the above random walk model, resulting in greater probabilities for the walker to walk into nodes with greater F values. In our implementation, we define the impact factor F_i of a web page (document node i) as a smoothed sum of its click counts as $F_i = 1 + \log(1 + \Sigma_j C_{ji})$, where C_{ji} is the weight between node j and i given by Q', and the impact factors for the other query nodes are set to be a constant 1. The new transition probability $P'(k|j)$ from node j to node k can be calculated by $P'(k|j) = Normalized(\frac{F_k}{\sum\limits_{i \in L_j^+} F_i} \times P(k|j))$, where L_j^+ is the set of outbound links of node j, $P(k|j)$ and F are calculated as earlier.

Besides using query graphs for document retrieval, we can also consider using user clicks as feedback. This additional query log data is useful for some web search algorithms. The third algorithm is a modified version of the implicit feedback model proposed in (Agichtein, Brill, and Dumais 2006). It merges the original rankings with the implicit feedbacks, in our case the user clicks. Previous literature has proposed methods that incorporate user behavior data to help to improve the order of retrieved documents. In this work, we implement a variant of (Agichtein, Brill, and Dumais 2006). Given a query q, the relevance score $S(d)$ for each document d is $\lambda \frac{1}{I_d+1} + (1-\lambda)\frac{1}{O_d+1}$ if the implicit feedback exists for d. Otherwise it is $\frac{1}{O_d+1}$. Here O_d represents the original rank of document d, I_d represents the implicit feedback rank in Q_{test}, and λ is a parameter to weigh the importance of the implicit feedback. In this approach, the original rank O_d is ranked by the order of click-through counts of document d for query q, according to Q'. I_d is the rank of d from Q_{Test} when the user makes a click. We empirically set $\lambda = 0.6$. Finally, the documents are ranked in the descending order of $S(d)$ scores for each query q. In addition, we generate a ground truth set of documents based on the actual clicking information in Q_{Test} to evaluate the document retrieval results obtained from the previous step. For each tested query in Q_{Test}, the corresponding ground truth contains a set of relevant documents (URLs), where relevant means that they have been clicked on in Q_{Test}. Actual search engines may choose to replace our approach in this step since they have more detailed data about users' online activity (for instance, the dwell time on each returned page) than we do.

Calculating Retrieval Effectiveness. We compare our retrieval results to the ground truth and evaluate the retrieval effectiveness using multiple IR metrics, including nDCG (normalized discounted cumulated gain) (Järvelin and Kekäläinen 2002), MAP (Mean Average Precision) (Robertson 2008), Precision, and Recall. Among them, nDCG at rank position 10 (nDCG@10) is the most widely used IR evaluation metric for web search in both commercial companies and academia. It measures the retrieval effectiveness for a ranked list of retrieved documents in the first ten results, which form a SERP that a searcher cares most about. We also use nDCG@10 as our predominant evaluation metric. For each query q in a query log Q, we can calculate its nDCG@10 as:

$$nDCG@10(q,D) = \{\sum_{i=1}^{10} \frac{rel_i}{log_2(i+1)}\}/\{\sum_{i=1}^{N_{Rel}} \frac{1}{log_2(i+1)}\} \tag{12}$$

where D is the ranked list of documents retrieved for query q by ranking algorithm R such that $D = R(q)$. rel_i is 1 when the i^{th} retrieved document $d_i \in D$ is relevant. Otherwise, it is 0. N_{Rel} is the smaller value of the total number of relevant documents for q and 10.

This work is the first to evaluate the utility of a query log using actual IR evaluation metrics. Our total utility function $U(Q)$ for a query log Q is:

$$U(Q) = \frac{1}{|Q|} \sum_{q \in Q} nDCG@10(q,D) \tag{13}$$

Note that not all queries in Q_{Test} can be found in the anonymized query log Q'. Most of the added queries in Q' from the query pool may not be included in Q_{Test} either. Therefore, when this utility function is used on the anonymized query log Q' to test on retrieval on Q_{Test}, we can only evaluate those queries that are at the intersection of both Q_{Test} and Q'. The size of the query set thus often varies.

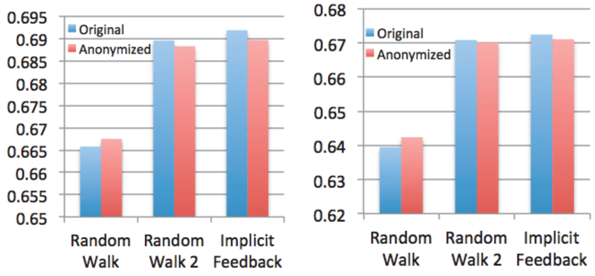
Experiments

This section presents our empirical results and analysis.

Experimental Setup

To evaluate the strengths and weaknesses of our approach, we use the released AOL query log data set (Adar 2007) for our experiments. It is a query log containing 36,389,567 search records, 10,154,742 unique queries and 19,442,629 clicks. At this stage, the AOL query log is the only available query log for privacy-related research like ours. Table 1 gives a sample of the original AOL query log.

As detailed in the previous section, we first partition the input query log into a training set Q and a test set Q_{Test}. Then we use our query log anonymization algorithm to generate the anonymized query log Q'. After that, we use the presented document retrieval algorithms to retrieve documents for queries in Q_{Test}. Finally, the utilities are calculated by comparing search results against the ground truth. During parameter tuning of the query log anonymization algorithm, we ran different combinations of the major parameters, including K, b, q_f, and c_f. To better control and compare the major parameters, we set constant values for some other parameters such as b_c and b_q.



(a) nDCG@10 (b) MAP

Figure 3: Utilities (nDCG@10 and MAP) of the anonymized log vs. the original log based on three algorithms.

Utility by Retrieval Effectiveness

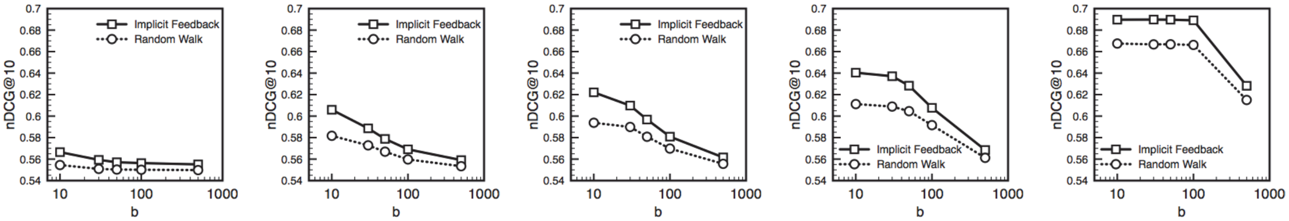
We begin by comparing retrieval results using the original query log (Q) and the anonymized query log (Q'). Figure 3 shows the utility across different retrieval evaluation metrics including nDCG@10 and MAP (Mean Average Precision). For each of the three algorithms we presented in the previous section, we run them on both the original log and an anonymized log with the following privacy settings: $\epsilon = 29.99$, query frequency threshold $K = 500$, and noise scale $b = 10$.

Results in Figure 3 indicate that our anonymized query log can produce comparable query effectiveness results to those of the un-anonymized version. Under certain circumstances, Q' can perform as well as the original non-private query log. This occurs when the noise scales b is much smaller than the query count threshold K. This means that the statistics in the released query logs are not influenced significantly by the added noise. These results confirm the utility level of the anonymized log generated by our framework.

Privacy-Utility Trade-off

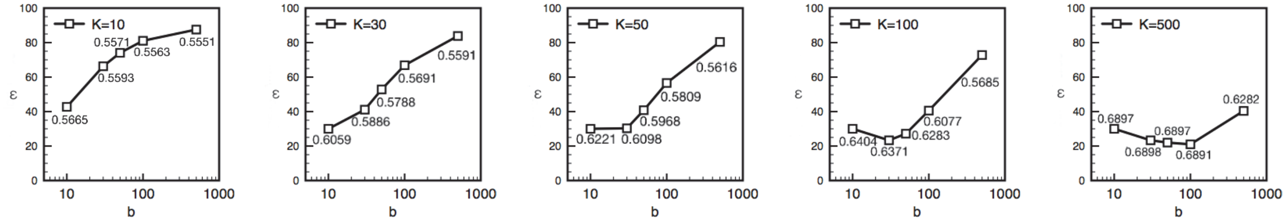
In this section, we consider the privacy-utility trade-off by adjusting different parameter settings to see when the retrieval performance decreases significantly. Among the three retrieval algorithms we have, two of them are based on random walk model and are similar to each other. Therefore, in this subsection, we focus our discussion on the regular random walk algorithm and the implicit feedback algorithm.

Figure 4 shows the document retrieval evaluation results using Q' with different K and b values. Each subgraph shows experiments using a different K value with K ranging from 10 to 500. Each data point on the subgraph represents the average of a set of 5-fold cross-validation results from the two retrieval algorithms (Implicit Feedback, and Random Walk algorithm). Within each subgraph, all the data points share the same q_f, c_f and K values. They also use the



(a) K=10 (b) K=30 (c) K=50 (d) K=100 (e) K=500

Figure 4: Relationships between noise scale b and utility $nDCG@10$, with K and other parameter values fixed.



(a) K=10 (b) K=30 (c) K=50 (d) K=100 (e) K=500

Figure 5: Relationships between b and privacy level ϵ. The data points are marked with their corresponding $nDCG@10$ scores using the Implicit Feedback algorithm.

same Q' size. Therefore, the results within each subgraph highlight the effect of different values of b. In general, as b increases, the utility $nDCG@10$ decreases. This matches intuition since we expect larger noise scales to reduce retrieval performance and cause decreased utility.

Another observation is that the utility score is less sensitive to the noise scale b when b is much smaller than K. Finally, we can see that the implicit feedback algorithm performs better than the random walk algorithm, but they share similar patterns of retrieval performance as b changes. Note, because different values of K lead to different sizes of data, comparing across subgraphs does not make sense.

Another experiment that reveals the privacy-utility trade-off focuses on the performance differences caused by q_f (query limitations from each user in Q) and c_f (clicking limitations from each user in Q). It is worth noting from Equation 2 that ϵ is very sensitive to q_f and c_f since it is linearly related to q_f and c_f. However, q_f and c_f also linearly control the actual size of query logs used in the experiments. Hence, to fairly understand the effect of changing these parameters, we need to ensure that each run has the same data size. We accomplish this by requiring that these experiments contain the same total number of queries and clicks used from the original query log Q. Intuitively, runs with smaller q_f and c_f values will have search data from more users while runs allowing more queries and clicks from each user will have fewer users in the anonymized query log.

Table 3 presents the evaluation results of this experiment, using the implicit feedback algorithm (IF), the random walk algorithm (RW) and the random walk algorithm with impact factor (RW-2). As we can see, while all the runs have the same number of total input queries and clicks, the evaluation results do not differ much from each other. However, we do observe that smaller q_f and c_f (input data obtained from more users) leads to smaller ϵ value (more privacy) and slightly smaller $nDCG@10$ scores (less utility), and vice versa. In other words, we also observe the privacy-utility trade-off with the change of q_f and c_f.

Table 3: Privacy-utility trade-off: the relationship between q_f,c_f and utility. All runs have the same number of total queries and clicks in Q_{clean}, with $K = b = 10$.

q_f	c_f	ϵ	IF	RW	RW-2
10	10	4.27	0.5884	0.5826	0.5891
50	50	21.36	0.5888	0.5828	0.5893
100	100	42.73	0.5903	0.5843	0.5909
150	150	64.09	0.5910	0.5850	0.5916
200	200	85.45	0.5920	0.5850	0.5923

Privacy Analysis

In this section, we analyze the privacy parameters and give recommendations to query log owners who want to use our framework to release and evaluate query logs. Figure 5 shows 25 different released query logs with different K and b values. The remaining parameters are fixed. They are organized in 5 subgraphs that show the b-ϵ relationships, each with different K values. In each subgraph, we label each point with the evaluated utility score ($nDCG@10$ from the implicit feedback algorithm). The figures show that the ϵ value is not always monotonically related to b. In graph 5(d) and 5(e), we can observe turning points with minimum ϵ values. These data points represent the smallest ϵ value we can achieve, i.e. the strongest privacy. We also notice that the utility at these points remains high. As b increases after the turning point, our performance decreases both regarding privacy (greater ϵ) and utility (smaller $nDCG@10$). Such turning points can be mathematically calculated from Equation 2. We get such turning points when $\frac{e^{1/b}}{p_g} = 1 + \frac{1}{2e^{(K-1)/b}-1}$. Base on these experiments, we recommend query log owners using Safelog should set parameters near these turning points. We also recommend using smaller b values (around $b = 10$) when K values are small (e.g., $K = 10$, 30, or 50). As K gets larger (e.g., $K = 100$ or 500), it is better to set b to be the same scale as K and close to the turning point. These settings should achieve the best combination of privacy and utility. Finally, if we choose q_f and c_f carefully, we can tune according to our needs. For example, smaller values of q_f and c_f lead to better privacy, while larger values of q_f and c_f lead to higher utility.

Exploring Web Mining

Query logs are used for some different web mining tasks. While testing the effectiveness of using an anonymized query log for this different task is outside the scope of this paper, we conduct a preliminary analysis on a simple web page clustering task. The goal of this task is to group n websites $W = \{w_1...w_n\}$ using query log click information from similar queries. We now sketch how to accomplish this and show that we get meaningful clusters using the anonymized query log Q'. We pause to mention that we purposefully do not conduct a complete analysis of this task. Instead, we demonstrate that web mining utility is still possible using Q' and that future work should explore more tasks from these anonymized query logs.

We consider a simple single-link hierarchical clustering algorithm (Guojun, Chaoqun, and Jianhong 2007) to cluster the websites (URLs) in Q' using the generated query-click graph. In the query-click graph, for each query q and website w, we define $C(q, w)$ as the number of clicks from query q to website w. We also define $C(w)$ to be the total number of clicks to website w. $QSet(w)$ is defined as the set of queries leading to clicks on website w. Then, for every website pair w_1 and w_2, we define their similarities as the ratio of clicks from common queries:

$$Sim(w_1, w_2) = \min\{\frac{\sum\limits_{q \in QS(w_1,w_2)} C(q, w_1)}{C(w_1)}, \frac{\sum\limits_{q \in QS(w_1,w_2)} C(q, w_2)}{C(w_2)}\} \tag{14}$$

where $QS(w_1, w_2) = QSet(w_1) \bigcap QSet(w_2)$. Once we compute the similarities, we use single-linkage clustering to cluster the websites. We empirically tested different hierarchical clustering methods and chose single link clustering for this task because of the sparsity of the query log click graph. The parameter setting for generating Q' were: query count threshold $K = 100$, query limit and click limit per user $q_f = c_f = 100$, all noise scales $b_x = 10$.

To evaluate the quality of the clusters, we clustered 1000 websites. The hierarchical clustering algorithm merges two clusters when the websites have a high similarity score: $Sim(w_1, w_2) \geq 0.5$. We hand label 10 clusters with class labels (Table 4 shows an example) and then measure *purity*.

Table 4: Some clustering results based on the anonymized query log.

(Cluster of Lyrics)	(Cluster of Lottery)
lyrics.astraweb.com	lottery.yahoo.com
www.lyricsfreak.com	www.flottery.com
www.lyrics.com	www.lotteryusa.com
www.azlyrics.com	www.flalottery.com
www.sing365.com	(Cluster of Banks)
www.musicsonglyrics.com	www.bankone.com
www.lyricsdownload.com	www.chase.com

To compute purity, we use the class label that is most frequent in the cluster and assumes that to be the correct class label. The accuracy is the number of correctly assigned website w_i divided by the total number of websites $|W|$. For the 10 clusters we hand labeled, there was a total of 59 websites in them. The purity was 0.76. This means that there were some websites that were put into the wrong clusters, but the majority were not. In other words, a meaningful structure for web mining can still be extracted from anonymized logs.

Conclusions

Given concerns about privacy, web search companies are hesitant to share web query logs. In this paper, we introduce Safelog - a framework for anonymizing and evaluating the utility and privacy of the anonymize log. To the best of our knowledge, we are the first to generate anonymized query logs that have been measured for utility on actual web search tasks. Our framework provides effective query log anonymization algorithms that place adequate privacy guards on those logs while simultaneously maintaining high retrieval utility. The experiments show that the proposed framework is very effective – a statistical significance test (two-tailed t-test, p $<$ 0.01) shows that popular web search algorithms using the anonymized logs perform comparable with those using logs before anonymization. In addition, our comparative experiments illustrate the privacy-utility tradeoff in query log release. In particular, the stricter the privacy standard we require, the lower the utility or usefulness of the released query log regarding web search. In this work, we show that the differentially private query log can be well supporting typical web search and mining tasks. We hope that it encourages web search engine companies to release logs for research purposes.

Acknowledgments

This research was supported by NSF grant CNS-1223825, NSF grant IIS-145374, and DARPA grant FA8750-14-2-0226. Any opinions, findings, conclusions, or recommendations expressed in this paper are of the authors, and do not necessarily reflect those of the sponsor.

References

Adar, E. 2007. User 4xxxxx9: Anonymizing query logs. In *Query Logs Workshop at the WWW'07*.

Agichtein, E.; Brill, E.; and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*.

Barbaro, M., and Zeller, T. Aug 2006. A face is exposed for AOL searcher no. 4417749. In *New York Times*.

Cai, F.; Liang, S.; and de Rijke, M. 2014. Personalized document re-ranking based on bayesian probabilistic matrix factorization. In *SIGIR '14*.

Carpineto, C., and Romano, G. 2013. Semantic search log k-anonymization with generalized k-cores of query concept graph. In *ECIR'13*.

Cheng, X.; Su, S.; Xu, S.; Tang, P.; and Li, Z. 2015. Differentially private maximal frequent sequence mining. *Comput. Secur.* 55(C):175–192.

Craswell, N., and Szummer, M. 2007. Random walks on the click graph. In *SIGIR '07*.

Davies, M. 2011. N-grams data from the corpus of contemporary american english (coca). *Downloaded from http://www.ngrams.info* 23:2012.

Diriye, A.; White, R.; Buscher, G.; and Dumais, S. 2012. Leaving so soon?: Understanding and predicting web search abandonment rationales. In *CIKM '12*.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06.

Dwork, C. 2008. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*. Springer. 1–19.

Fan, L.; Bonomi, L.; Xiong, L.; and Sunderam, V. 2014. Monitoring web browsing behavior with differential privacy. In *WWW '14*.

Feild, H. A.; Allan, J.; and Glatt, J. 2011. Crowdlogging: Distributed, private, and anonymous search logging. In *SIGIR '11*.

Gotz, M.; Machanavajjhala, A.; Wang, G.; Xiao, X.; and Gehrke, J. 2012. Publishing search logs – a comparative study of privacy guarantees. *IEEE Trans. on Knowl. and Data Eng.* 24(3):520–532.

Guojun, G.; Chaoqun, M.; and Jianhong, W. 2007. Data clustering: theory, algorithms, and applications. *ASA-SIAM Series on Statistics and Applied Probability*.

Hong, Y.; He, X.; Vaidya, J.; Adam, N.; and Atluri, V. 2009. Effective anonymization of query logs. In *CIKM '09*.

Järvelin, K., and Kekäläinen, J. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.* 20(4):422–446.

Jones, R.; Kumar, R.; Pang, B.; and Tomkins, A. 2007. "i know what you did last summer": Query logs and user privacy. In *CIKM '07*.

Jones, R.; Kumar, R.; Pang, B.; and Tomkins, A. 2008. Vanity fair: Privacy in querylog bundles. In *CIKM '08*.

Korolova, A.; Kenthapadi, K.; Mishra, N.; and Ntoulas, A. 2009. Releasing search queries and clicks privately. In *WWW '09*.

Lee, J., and Clifton, C. W. 2014. Top-k frequent itemsets via differentially private fp-trees. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 931–940. New York, NY, USA: ACM.

Luo, J.; Zhang, S.; and Yang, H. 2014. Win-win search: Dual-agent stochastic game in session search. In *SIGIR '14*.

Radlinski, F., and Joachims, T. 2005. Query chains: Learning to rank from implicit feedback. In *KDD '05*.

Robertson, S. 2008. A new interpretation of average precision. In *SIGIR '08*.

Shokouhi, M.; White, R. W.; Bennett, P.; and Radlinski, F. 2013. Fighting search engine amnesia: Reranking repeated results. In *SIGIR '13*.

Tran, G.; Turk, A.; Cambazoglu, B. B.; and Nejdl, W. 2015. A random walk model for optimization of search impact in web frontier ranking. In *SIGIR '15*.

Zhang, S.; Luo, J.; and Yang, H. 2014. A pomdp model for content-free document re-ranking. In *SIGIR '14*.