Deriving Differentially Private Session Logs for Query Suggestion

Sicong Zhang
Department of Computer Science
Washington, D.C., USA
sz303@georgetown.edu

Grace Hui Yang
Department of Computer Science
Washington, D.C., USA
huiyang@cs.georgetown.edu

ABSTRACT

Query logs are valuable resources for Information Retrieval (IR) research. However, the huge concern about user privacy has become an obstacle preventing data from being released and interfering with the advance of IR. It is understandable but still quite disappointing. Recent privacy research has begun to address the problem by anonymizing queries in query logs. In that, each query and its associated user actions are treated as one block for anonymization; each block is independent from each other. It might be sufficient to support ad-hoc retrieval that handles queries independently but will be not adequate for more complex IR tasks that require the knowledge of query sequences. In this paper, we tackle this challenge by keeping session information in differentially privately anonymized logs so that an anonymized log is able to support IR tasks, such as query suggestion and session search, that need query sequence information as well. We also provide analysis on how to achieve a proper balance between privacy and search utility.

KEYWORDS

Differential Privacy; Query Log Anonymization; Query Suggestion

1 INTRODUCTION

Query logs are essential research materials for understanding how users interact with the search engine. However, the large and 'real' query logs are not readily accessible by the general research community due to privacy concerns. The dominance of these research materials by only a few commercial companies might have been negatively impacting the research field as a whole. We suspect an increasing split in the Information Retrieval (IR) community – academic researchers who have no access to query logs and could not conduct related research vs. industrial researchers who have the data but miss the opportunities to learn more diversified ideas from their academic colleagues. The authors of this paper believe that this split could be one of the reasons that premium IR conferences such as SIGIR are experiencing a decline.[1] We probably won't be able to make changes to the situation in one day, but we do hope the issue of privacy in IR could be alleviated at least from a technical point of view. The authors of this paper are highly motivated to propose query log anonymization methods to enable data release for research in IR.

[1]http://sigir.org/files/forum/2016J/p001.pdf

ICTIR '17, October 01-04, 2017, Amsterdam, Netherlands
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4490-6/17/10…$15.00
https://doi.org/.1145/3121050.3121076

The principle challenge of privacy protection is to safeguard the data from being identifiable at the individual level – which is *data anonymization*. Anonymization is a lossy process. Unavoidably, an anonymized dataset would become less useful than the original one. Therefore, another challenge of privacy protection is to preserve the utility of the anonymized data as much as we can.

In the early years, most query log anonymization techniques were quite straightforward [1, 11, 19]. Those approaches had no proved privacy guarantee and the anonymized information could be easily re-identified. Unfortunately, however, those poorly-proved methods are still in use during recent data releases. For instance, in 2014, Yandex released a detailed query log containing over 167 million search records from 5 million users using only hash coding and data deletion.[2] The resulting query log contains no meaningful natural language words but can still be easily deciphered if one has knowledge of popular website frequencies. A theoretically and practically sound way to anonymize query logs is very much needed to remove us from such high risks.

More recently, a technique called *k-anonymity* has been applied to query log anonymization [9, 18]. The basic idea is to hide sensitive information among many (say k) similar copies of data. It is like *concealing a drop in the sea.* However, a serious shortcoming exists in k-anonymity, which is that it assumes knowing what knowledge an adversary has access to. In the real world, however, such assumption could never hold because an adversary could always gather extra knowledge from unknown sources and use that to decipher the k-anonymized data. Therefore, it is really not safe to use k-anonymity for data protection.

Until lately, researchers have explored stronger privacy protection mechanisms. The state-of-the-art method is Differential Privacy (DP) [12]. DP is effective in anonymizing statistics about a dataset as well as is able to provide mathematically proved privacy guarantee. By adding noise to sample statistics in a dataset, DP creates disturbed (anonymized) statistics that no one could tell whether one exists in the dataset or not. In this way, DP hides the information of each individual. The idea is that if no one could tell a user's existence, it would be even more impossible for them to find the user out. It is like what is described in the *Platform Sutra – if fundamentally there is not a single thing, where could any dust be attracted?* DP is perhaps the strictest way to protect data privacy since the data 'seems' to no longer exist. Due to its strong privacy protection, there is usually no need for DP to make assumptions

[2]https://www.kaggle.com/c/yandex-personalized-web-search-challenge/data

Table 1: **Toy example for Differential Privacy (DP): After anonymization, from an anonymized sum, it is difficult to tell whether Carol is in Dataset 1 and/or Dataset 2.**

	Dataset 1	Dataset 2
Raw Data	Alice has 5 apples Bob has 4 apples Carol has 2 apples	Alice has 5 apples Bob has 4 apples
Sum of apples	5+4+2=11	5+4=9
Anonymized Sum	11+Noise=10	9+Noise =10

on what knowledge or what method an adversary would use to attack a dataset. The no requirement of assumptions makes DP very plausible to be used in practice.

Table 1 shows a toy example and explains how DP works. Suppose we would like to release a database about users' possession of apples. We are not going to release the raw data. Instead, we would like to release a summary statistic to the public. Would there be any privacy concern when we release sum as the statistic? The answer is 'yes'. Suppose we will release either Dataset 1 or Dataset 2, which are two databases differ by exactly one user, Carol. The summary statistic, the sum of apples, for the two datasets will be different by 2, that is what is contributed by Carol (11-9=2). If an adversary happens to know that (a) Carol has 2 apples and (b) everybody else has either 4 or 5 apples, then it is easy to identify that Carol is in Dataset 1 and not in Dataset 2. The re-identification can be done by exausitively calculating a few possible decompositions for the released statistic. In this case, the only possible decomposition for 11 is 5+4+2 and for 9 is 5+4. The adversary can thus identify which dataset Carol is in since Carol must be included in Dataset 1 and excluded from Dataset 2. Once knowing where Carol is, it would become fairly easy to figure out other information about Carol by the adversary. To obscure an individual's identity, DP adds randomized noise to the summary statistic. The anonymized data then follows a distribution, whose mean equals to the original mean, but the end results would not be distinguishable between datasets with or without Carol. In other words, after applying DP, the adversary would not be able to figure out whether or not Carol is in a given dataset based on the released statistics.

Researchers have made progresses in applying DP in query log anonymization [14, 22, 29]. There are two main limitations of the current approaches. First, most approaches simply measure the utility of anonymized logs by the percentage of what is remaining, instead of evaluating through actual IR tasks and IR effectiveness measures [22]. In this study, we find that percentage of the kept data and real IR utility could contradict each other. We argue that we should use the latter. Second, existing work only takes care of single record privacy protection [29]. This simplification is not sufficient for complex IR tasks that require the use of sessions, a particular form of sequential data, from a query log.

A major challenge of session log anonymization is the sparsity of re-occurring sessions. The sparsity comes from the large number of possible query sequences. From the perspective of privacy research, the sparsity of sessions would require a greater scale of noises to be added and would make the privacy protection harder to stay under control. From the perspective of IR utility, the sparsity of sessions also makes it more complicated to distinguish higher frequency sessions, which may be more valuable to support IR applications, from the other sessions.

In this paper, we tackle the challenge and propose a releasing mechanism to produce a DP-protected query log with both click-through data and session data. We show that our anonymized query log can be used to support complex IR application such as query suggestion and report the remaining IR utility of the anonymized log. We wish this work can contribute to our research community by better enabling data release for IR studies.

2 RELATED WORK

Naive Privacy Techniques: In 2006, an outbreak of tremendous privacy concerns was triggered by that many users were re-identified from a query log released by America Online (AOL) [3]. The AOL log was "anonymized" by hash coding the ID of each user, which was soon proved a poor mechanism. Since then, privacy-enhancing techniques [1, 11, 14, 20] have been tried on query logs. Early attempts include *log deletion, hashing queries, identifier deletion, hashing identifiers, scrubbing query content, deleting infrequent queries* and *shortening sessions* [11]. Those methods all tried to directly modify or remove individual data entries. However, there would always be some other traits left to re-identify an individual. It is like you may still recognize a friend who stands in front of you, even if he covers his name tag (hashing identifier) or wears a mask (scrubbing query content). Researchers have reached a consensus that those early techniques do not work [1, 14, 20]. Jones et al. [20] showed that after removing unique terms from a query log, personal information could still be re-identified with a simple classifier at an accuracy as high as 97.5%. We urge our research community or anyone who intends to release a query log to stop using those naive techniques.

K-Anonymity: K-Anonymity [9, 25] has been a popular privacy protection technique since 2002. The main idea is that any released record for an individual can not be distinguished from at least k-1 other individuals whose information are released too [25]. In the query log anonymization scenario, to satisfy k-anonymity, an anonymization algorithm would only release query click records appearing at least k times in the original query log [9]. Here k is usually the cut-off threshold of some frequency. However, the privacy of k-anonymity is not strong because it depends on assumptions made about an adversary [25]. To develop data protection mechanisms with proved privacy guarantee, privacy research has moved on to differential privacy.

Differential Privacy: Differential privacy [13, 14, 22, 23, 26, 28, 29] is the state-of-art privacy protection mechanism especially for statistical data release. The key idea of differential privacy is to make the released statistics affected very limited by adding or removing a single record or a single user. Whether achieving record level DP or user-level DP depends on the neighboring dataset's definition. In this paper, we implement user-level DP. Recent work [14, 22, 28] has been proposed to use histogram based DP algorithms for query log anonymization. Gotz et al. [14] and Korolova et al. [22] proposed to release queries and clicks based on their frequencies. They were among the first to release queries as natural language words instead of as hash codes. Their work achieved (ϵ, δ)-differential privacy. However, the utility of the released query log was evaluated by the quantity of queries that can be released and how similar the released statistics are to the original statistics. They showed that more than 10% of the search volume and 0.75% distinct queries from the original log could be privately released. Zhang et al. [28] proposed another algorithm that could achieve ϵ-differential privacy with greater ϵ and zero δ. The work used web document retrieval to evaluate the utility of the released log. However, their use of an external query pool may raise other privacy concerns. The existing DP work focuses on protecting single queries. In this paper, we propose an (ϵ, δ)-differential privacy method for anonymizing a session of multiple queries in sequences.

Data Sequence Release: Frequent sequence mining [4, 7, 10, 24, 27] is relevant to our work because it studies how to anonymize sequences of data. Their techniques can be grouped by types of sequences being released – consecutive subsequence mining [7, 10], unconstrained subsequence mining [27], and frequent itemset mining [24]. It is worthy noting that these techniques cannot be directly applied to query log anonymization. The reason lies deep in the difference between IR and data mining (DM)/database (DB). IR handles free text in natural languages, which can be considered as an infinite domain of data. On the contrary, DM and DB handle structured data, which is generated from a limited domain, also called a limited or controlled vocabulary. Most frequent sequence mining approaches would function at a very high cost for even a small controlled vocabulary, which makes it inapplicable on free text tasks. To the best of our knowledge, this paper is the first to study how session data can be released anonymously and can be useful to support IR tasks.

Query Suggestion: Query Suggestion [16] is an important task in IR. Although under specific settings, query suggestion can be done in the absence of query logs [5], most query suggestion approaches utilize query logs to suggest queries [2, 6, 8, 17, 21]. There are two main types of log-based query suggestion approaches. The first type cluster queries based on a query-click bipartite graph. For instance, Baeza-Yates et al. [2] proposed a method based on query clustering which groups semantically similar queries together. The click-through data in the log is used to represent and cluster the queries. The other approaches employ query sequences to predict the follow-up queries. For instance, Boldi et al. [6] utilized a query-flow graph to generate query suggestion results. Here, the most valuable pieces of information for query suggestion are click-through data and query sequences. In this paper, we anonymize both pieces of information in order to support query suggestion.

3 ANONYMIZATION ALGORITHM

In this section, we first introduce the related definitions of DP. Then we present our DP query log anonymization algorithm, followed by a brief proof of privacy.

3.1 Differential Privacy and Other Definitions

Neighboring is an important concept upon which differential privacy is defined. We therefore start from defining what neighboring datasets are.

Definition 1: Neighboring. Two query logs, or more generally two datasets, Q_1 and Q_2, are said to be neighboring to each other if they differ by at most one user.

The toy example we presented in Table 1 shows two neighboring datasets. The only difference between them is the information from one person, Carol. When we define differential privacy later, any possible pairs of neighboring datasets could be under consideration. It means that the datasets could also differ by Alice or by Bob.

Definition 2: Differential Privacy. A randomized query log anonymization algorithm A satisfies differential privacy, or more specifically (ϵ, δ)-differential privacy, iff for all neighboring query logs Q_1 and Q_2 and for all possible output logs Q', the following inequality holds:

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

where A is the randomized anonymization algorithm that inputs an original query log and outputs an anonymized query log Q'. ϵ and δ indicate privacy levels of A and control the probabilities of getting certain outputs from the neighboring inputs.

Based on Eq. 1, the smaller the values of ϵ and δ, the stronger the privacy guarantee. The ranges of ϵ and δ can be $0 \leq \epsilon \leq \infty$ and $0 \leq \delta \leq 1$. However, we usually expect ϵ and δ to be much smaller than their upper bounds. In general there is no hard rule for selecting their values and proper settings vary quite a lot depending on the actual applications. However, usually we consider an ϵ no more than 10 and a δ less or around 1/#of released users.

According to Eq. 1, the values of ϵ and δ directly affect how different two neighboring query logs are after applying the anonymization algorithm A. Therefore it is necessary to quantitively define the difference between two neighboring input query logs Q_1 and Q_2. The following defines *sensitivity*:

Definition 3: Sensitivity. Given a function f that takes a query log Q as the inputs and produces a numerical vector as the output. The sensitivity of f is denoted as Δf:

$$\Delta f = \max_{\forall \text{ neighboring } Q_1, Q_2} ||f(Q_1) - f(Q_2)||_1 \qquad (2)$$

where $||.||_1$ is the l_1 norm. The maximum is taken over all pairs of neighboring query logs Q_1 and Q_2. In the scenario of query log anonymization, the numerical vector output, $f(Q)$, is a vector of the raw count statistics generated from query log Q. The sensitivity value Δf indicates the maximum overall statistical difference between the two neighboring inputs Q_1 and Q_2, which would largely influence the privacy levels, i.e. the values of ϵ and δ.

3.2 Query Log Anonymization Algorithm

Our query log anonymization algorithm releases two types of data: query sequences in a session and click-throughs. The nonymization algorithm also consists of two parts, $A_{Session}$ and A_{Click}, to produce the two types of data. Both parts take the original query log Q as the input and satisfy the (ϵ, δ)-differential privacy. $A_{Session}$ releases frequent search sessions as the output, while A_{Click} releases frequent query click-through data as the output. We merge their outputs and release both.

3.2.1 Releasing Session Data. Because of the property of nature of natural language, the total number of unique search queries could be infinite. Exact match of the two search sessions is thus rare because sessions are ordered sequences of natural language queries. As we know that it is difficult to protect low frequency data from being re-identified. To address this challenge, we propose to increase the session and query counts by adding all non-trivial subsequences in sessions.

We propose the $A_{Session}$ algorithm to release search sessions with (ϵ, δ)-differential privacy. Each non-trivial subsequences of the original sessions would contribute a count to reduce the sparsity of sessions. We add noise to the session frequency by applying the Laplace mechanism [14, 22].

Algorithm $A_{Session}$ takes in the following inputs. The original query log Q, the session timeout threshold T_{Gap}, the max number of sessions per user l_s , the max number of queries per session l_q, Laplace noise scale b, and the session frequency cut-off threshold K. The algorithm is described as the following:

1) Session segmentation. We define each session S as an ordered sequence of search queries:

$$S = [q_1, q_2, ..., q_{|S|}] \tag{3}$$

where $|S|$ is the number of queries in S.

We segment the sessions in Q based on either i) if they are from two different users, or ii) if the timestamp difference between these two queries are greater than the session timeout threshold T_{Gap}. Practically, we take 30 minutes as the timeout threshold according to previous work in session search [15]. Then we transfer Q into Q_{seg}, which is a set of sessions:

$$Q_{seg} = \{S_1, S_2, ..., S_{|Q_{seg}|}\} \tag{4}$$

Without loss of generality, we only keep non-trivial sessions that consist of at least two queries. “Sessions" with only a single query are excluded from Q_{seg}.

2) Subsequence mining. To increase the number of repeated sessions, we take into consideration all non-trivial subsequences within a session and treat them as sessions too. The relative orders from the original session are reserved. with the permission to have some original queries missing in the new subsequences. That is to say, each subsequence of a session $S = [q_1, q_2, ..., q_{|S|}]$ is a sequence $S' = [q'_1, ...q'_{|S'|}]$ defined by $q'_t = q_{n_t}$, where the indices $n_1 < n_2 < ... < n_{|S'|}$ is monotonically increasing and $|S'| >= 2$. For example, if the original session is (q_1, q_2, q_3, q_4), our algorithm adds one count to each of the following 11 sessions:

$$\begin{gathered}(q_1, q_2, q_3, q_4), (q_1, q_2, q_3), (q_1, q_2, q_4), (q_1, q_3, q_4), (q_2, q_3, q_4)\\ (q_1, q_2), (q_1, q_3), (q_1, q_4), (q_2, q_3), (q_2, q_4), (q_3, q_4)\end{gathered} \tag{5}$$

Hence, the search sessions, or query sequences, may get greater frequency of occurrence from the same raw dataset.

3) Sensitivity control. In order to make sure the presence or absence of each individual user would not make impact too much on the statistics that we release, the sensitivity Δf of $A_{session}$ is controlled by adjusting the max number of sessions per user l_s and the max number of queries per session l_q. Sessions longer than l_q queries are trimmed down to only the first l_q queries. As a result, the sensitivity is kept as $\Delta f = l_s \times (2^{l_q} - 1 - l_q)$. The proof can be found in Section 3.3.

4) Session release decision. We denote the session counts for a session S after subsequence mining as $C(S)$. Then we apply the Laplace mechanism on the session counts by adding i.i.d. noise to each count. We release session S iff. $C(S) + noise > K$, where $noise \sim Lap(0, b)$, and K is the session frequency cut-off threshold. The decision of whether to release session S is made based on:

$$\begin{cases} \text{if } C(S) + Lap(0, b) > K, & \text{Release session } S \\ \text{otherwise,} & \text{Do not release } S \end{cases}$$

5) Session count release. For all session S that we have decided to release, their counts generated from the previous step form a biased sample since they are all selected because their values are greater than K. However, we need to make sure the actual released counts still follow an unbiased Laplace distribution $Lap(C(S), b)$. We therefore need to do the sampling again and release the sessions together with their perturbed count $C(S) + Lap(0, b)$. The released perturbed frequency count $C(S) + Lap(0, b)$ follows the distribution of $Lap(C(S), b)$, which becomes the Laplace Mechanism [13] for Differential Privacy.

6) Output: A set of frequent sessions along with their corresponding counts, $\{(S, C(S))\}$, while each session S is in the form of an ordered sequence of free text queries. Table 2 shows an example outputted by $A_{Session}$.

3.2.2 Releasing click-through data. We also release click-through data, i.e. a query and the urls that a user clicked for the query. Algorithm A_{Click} releases the click-through data as a tuple of query, clicked document and the count for the pair, $[q, d, c(q, d)]$. The released data satisfies the (ϵ, δ)-differential privacy. The count $c(q, d)$ is the frequency after adding Laplacian noise for a query-URL pair (q, d).

The A_{Click} algorithm takes in the following inputs: the query log Q, query click records limit per user l, the Laplace noise scale b (the same as in $A_{Session}$), and the frequency threshold K (the same as in $A_{Session}$). Note that both $A_{session}$ and A_{Click} need to share the same privacy parameter settings b and K. It is because the overall privacy guarantee is bottlenecked by the algorithm that has the weaker differential privacy guarantee.

The A_{Click} algorithm releases the click-through data to assist session data release using the following steps:

1) Sensitivity control for clicks. For each user in Q, we only keep the first l click records for each user and ignore the rest from the same user to make sure that data from any user won't contribute too much to the overall frequency statistics.

2) Query-clickthrough release decision. This is similar to step 4 in $A_{Session}$. We first count the total number of occurrences for each query-clickthrough pair as $C(q, d)$. We then decide to release a pair (q, d) iff. $C(q, d) + noise > K$, where $noise \sim Lap(0, b)$ and K is the frequency cut-off threshold.

3) Release query-clickthrough tuples. This is similar to step 5 in $A_{Session}$. If a (q,d) pair has been decided to be released at the previous step, we perform a sampling again from the Laplacian distribution for added noise and release a 3-tuple $[q, d, C(q, d) + noise]$ where the new i.i.d. $noise \sim Lap(0, b)$; otherwise, we won't release anything for the (q, d) pair.

4) Output: A set of query-clickthrough tuples in the form of (Query q, Document d, Fuzzed Count $C(q, d) + noise$). Table 3 shows an example outputted by A_{Click}.

3.3 Proof of Privacy

According to the definition of (ϵ, δ)-differential privacy, a query log anonymization algorithm A should satisfy the following two

Table 2: Session output example.

Session 1	Session 2
q_1=daily record morristown nj q_2=star ledger newark nj q_3=google	q_1=ny lottery q_2=pa lottery q_3=nj lottery q_4=ny lottery
Counts: 11	Counts: 16

Table 3: Click-through output example.

Query	Clicked URL	Counts
weather	http://www.weather.com	4190
weather	http://weather.yahoo.com	1035
aol weather	http://weather.aol.com	30
aol weather	http://aolsvc.weather.aol.com	16

inequalities for all neighboring query logs Q_1 and Q_2 in order to achieve DP.

$$P[A(Q_1) \in \hat{Q}] \leq \alpha P[A(Q_2) \in \hat{Q}] + \delta \quad (6)$$

$$P[A(Q_2) \in \hat{Q}] \leq \alpha P[A(Q_1) \in \hat{Q}] + \delta \quad (7)$$

A major difference between our algorithm $A_{Session}$ and previous DP algorithms [14, 22, 29] is that we consider the query sequence of search sessions, rather than individual queries or click-throughs as the unit to be counted and to be protected. However, a common mechanism is shared by all the work, which is achieving the requirements of (ϵ, δ)-differential privacy (Equa. 6 and 7) by adding i.i.d noise from the Laplace distribution on the data. A proof of the mechanism can be found in the *Lemma 1.* of Section 5.1 of Korolova et al. [22]. It shows that adding Laplacian noise would be able to achieve $(d \cdot ln(\alpha), \delta_{alg})$-differential privacy, where $\alpha = Max\{e^{1/b}, 1 + \frac{1}{2e^{(K-1)/b}-1}\}$, $\delta_{alg} = 0.5d \cdot exp(\frac{d-K}{b})$.

Now let's evaluate the ϵ and δ values in our algorithm. In [22], d is the max number of queries per user and equals the sensitivity in their algorithm. That is to say, $d = \Delta f$ [22]. In our algorithm, the sensitivity Δf can be greater than the maximum of sessions per user l_s because the subsequences of original sessions also contribute to the sensitivity value. If we consider each of the Δf counts caused by a user as an individual record, the privacy guarantee of the Step 4 in $A_{Session}$ would be equivalent to $d = \Delta f$ as in Lemma 1. Hence the Step 1 to Step 4 (to decide which session may be released) of $A_{Session}$ satisfies $(\Delta f \cdot ln(\alpha), \delta_{alg})$-differential privacy, where $\alpha = Max\{e^{1/b}, 1 + \frac{1}{2e^{(K-1)/b}-1}\}$, $\delta_{alg} = 0.5\Delta f \cdot exp(\frac{\Delta f - K}{b})$. Step 5 of $A_{Session}$ is a standard procedure in differential privacy [13]. Such released session counts could be used to weigh query transitions in a session, which is helpful for the IR algorithms. This step itself achieves $(\Delta f/b, 0)$ differential privacy. Therefore, the overall $A_{Session}$ algorithm achieves $(\Delta f \cdot (ln(\alpha) + 1/b), \delta_{alg})$-differential privacy, while α and δ_{alg} are defined as earlier.

Next, we need to calculate the exact value of Δf in order to finalize the privacy level ϵ and δ. Sensitivity Δf is defined as the maximum difference of the statistics could be made by the data generated from one user. In the worst case, the particular user differing between two neighboring datasets may issue l_s sessions, and each session may contain at most l_q queries. Given a session containing l_q queries, the amount of subsequences containing at least two queries is 2^{l_q} (total subsequences) -1 (the empty sequence) - l_q (subsequences containing only 1 query). Hence each input session with l_q queries contributes a count by at most $2^{l_q} - 1 - l_q$ sessions. Therefore, the overall sensitivity of our algorithm is:

$$\Delta f = l_s \times (2^{l_q} - 1 - l_q) \quad (8)$$

By plugging in the sensitivity value Δf into $\epsilon = \Delta f \cdot (ln(\alpha) + 1/b)$ and $\delta = 0.5\Delta f \cdot exp(\frac{\Delta f - K}{b})$, our session release approach can achieve (ϵ, δ)-differential privacy while ϵ and δ are:

$$\epsilon = l_s(2^{l_q} - 1 - l_q) \cdot (ln(Max\{e^{1/b}, 1 + \frac{1}{2e^{(K-1)/b} - 1}\}) + 1/b)$$
$$\delta = 0.5 l_s(2^{l_q} - 1 - l_q) \cdot exp(\frac{l_s(2^{l_q} - 1 - l_q) - K}{b}) \quad (9)$$

4 UTILITY MEASUREMENT

Data utility should be evaluated task-dependently. However, most prior work simply measures the utility by how much data is kept. In our experiments (Section 5), we reveal that amount of kept data and the actual task-dependent utility do not agree. In this research, we a real IR task – query suggestio – and classic IR evaluation metrics to measures the utility of a query log after anonymization.

4.1 The Task of Query Suggestion

Query suggestion is a popular IR task. The goal of the task is to predict the next search query that a user is going to write. Given a session S with $n + t$ queries, $S = [q_1, q_2, ..., q_{n-1}, q_n, q_{n+1}, ..., q_{n+t}]$ the task of query suggestion is to generate a ranked list of suggested queries $\{q'_1, ..., q'_m\}$ as the candidates of the next query after q_n. For evaluation purpose, we use the queries that are after q_n in the same session and are generated by the same user as the ground truth. That is, $Truth(q_n) = \{q_{n+1}, ..., q_{n+t}\}$. The results can then be evaluated by comparing between the generated ranked list $\{q'_1, ..., q'_m\}$ and the ground truth set $Truth(q_n)$.

4.2 Query Suggestion Using Anonymized logs

We build two graphs G_s and G from the anonymized log Q' to support query suggestion. We use the first graph, a query-flow graph $G_s = (V_s, E_s)$, to organize queries in sessions. We use the second graph, a query-URL bipartite graph $G = (V_q, V_d, E)$, to organize relations between queries and URLs in anonymized click-through data.

With the help of the anonymized session information, we are able to create a query-flow graph as in Boldi et al. [6]. The query-flow graph G_s organizes the ordered query transitions from the query sequences in Q'. In particular, $G_s = (V_s, E_s)$. V_s contains the set of query vertex in the graph and E_s is the set of edges connecting queries that have occurred adjacently. In G_s, we denote $e(q_i, q_j)$ as the edge weight between the transition from q_i to q_j, which is number of co-occurrences of q_i and q_j in the anonymized session log. Note that q_j is any query that appears after q_i and is not restricted to be the query immediately after q_i in the session. We also denote $d(q_i)$ as the out-degree of q_i. Then the probability of q_j

following q_i in the same session can be calculated as $e(q_i, q_j)/d(q_i)$, if only based on the session data.

We also use the query click-through data. We organize queries and their corresponding click-through URLs into a query-URL bipartite graph $G = (V_q, V_d, E)$. V_q is the set of query nodes, V_d is the set of document (URL) nodes, and E is the set of weighted edges in G. According to this bipartite graph, we represent each query $q \in V_q$ as a vector of weighted documents $\overrightarrow{q}$. Then we calculate the similarities between any two queries q_i and q_j by their normalized dot product $\overrightarrow{q_i} \cdot \overrightarrow{q_j} / (|\overrightarrow{q_i}| \cdot |\overrightarrow{q_j}|)$. We use a variation of a state-of-the-art query suggestion approach [8] to quantify the similarities between the queries. The difference is that we generate a ranked list of relevant queries for each query q, rather than allocating queries into clusters [8]. The ranked lists of the relevant queries is equivalent to the results generated based on the Euclidean distance between the normalized feature vectors as in [8] according to the geometric properties of the vector space.

Finally, we combine the two scores from both G_s and G. The overall probability of having the candidate query q_j follow q_i is calculated as:

$$P(q_i, q_j) = \lambda \frac{\overrightarrow{q_i} \cdot \overrightarrow{q_j}}{|\overrightarrow{q_i}| \cdot |\overrightarrow{q_j}|} + (1 - \lambda) \frac{e(q_i, q_j)}{d(q_i)} \tag{10}$$

where λ is a parameter to control the value contributed between G and G_s.

4.3 Using Classic IR Metrics

In this paper, we use classic IR metrics Precision and Recall to evaluate the utility for query suggestion. In particular, we report Precision@5 and Recall@5:

$$Precision@5 = \frac{1}{5} \sum_{i=1}^{5} Hit(i);\ Recall@5 = \frac{1}{t} \sum_{i=1}^{5} Hit(i) \tag{11}$$

$$Hit(i) = \begin{cases} 1, & \text{if } q'_i \in Truth(q_n). \\ 0, & otherwise. \end{cases}$$

where $t = |Truth(q_n)|$, $Hit(i)$ shows whether the i^{th} predicted query q'_i hits the ground truth, $1 \le i \le 5$.

5 EXPERIMENTS

We evaluate our algorithms on the 2006 AOL dataset. The entire query log contains 36,389,567 search records. In total, there are 10,154,742 unique queries and 19,442,629 clickthrough records from 657,426 unique users over three months. We use nine-tenths of the query log as the original log Q to be anonymized, and reserve one-tenth of the data as the test set Q_{Test} for evaluating the IR applications. In the experiments, we compare a few query log anonymization schemes and use the anonymized logs Q' generated from each of them to test on the query suggestion task.

5.1 Anonymized Methods to Compare

The logs used in our experiments include the Original, *KA* (logs anonymized by k-anonymity), DP_C (logs anonymized by differential privacy, clickthrough data only), and DP_S (logs anonymized by differential privacy, containing both session and clickthrough data). The details of them are described as follows:

Table 4: Privacy levels ϵ and δ for typical $A_{Session}$ runs.

Detail Parameters in DP$_S$	ϵ	δ
b=1, K=10, T_{Gap}=30, l_s=1, l_q=3	ϵ=8.00	δ = 4.95 * 10^{-3}
b=1, K=20, T_{Gap}=30, l_s=1, l_q=3	ϵ=8.00	δ = 2.25 * 10^{-7}
b=1, K=30, T_{Gap}=30, l_s=1, l_q=3	ϵ=8.00	δ = 1.02 * 10^{-11}
b=3, K=20, T_{Gap}=30, l_s=1, l_q=3	ϵ=2.67	δ = 9.66 * 10^{-3}
b=3, K=30, T_{Gap}=30, l_s=1, l_q=3	ϵ=2.67	δ = 3.44 * 10^{-4}
b=1, K=20, T_{Gap}=30, l_s=1, l_q=4	ϵ=22.00	δ = 6.79 * 10^{-4}
b=2, K=30, T_{Gap}=30, l_s=1, l_q=4	ϵ=11.00	δ = 4.12 * 10^{-4}
b=1, K=20, T_{Gap}=30, l_s=2, l_q=3	ϵ=16.00	δ = 2.46 * 10^{-5}
b=2, K=30, T_{Gap}=30, l_s=2, l_q=3	ϵ=8.00	δ = 6.68 * 10^{-5}

Table 5: Query Suggestion results using different query logs

Run	Precision@5	Recall@5	# of Evaluated Sessions
Original	0.0421	0.1402	**18,475**
KA	0.0693	0.2312	9,494
DP$_C$	0.1133	0.3891	4,144
DP$_S$	**0.1139**	**0.3911**	4,119

- Original: The original query log Q without anonymization.
- KA(K): The query log anonymized by the k-anonymity [25]. This log contains frequent clickthrough data from the original log while preserving certain privacy with k-anonymity. Major steps of the k-anonymity query log anonymization algorithm are as follow:
  (1) Input: a query log Q, query clickthrough frequency threshold K.
  (2) Count the number of users who formulates query q and clicks document d as $c(q, d)$.
  (3) Release all tuples $[q, d, c(q, d)]$ iff. $c(q, d) > K$, where K is the frequency cut-off threshold.
  (4) Output: A set of tuples in the form of [Query q, Document d, User Count c(q,d)].
- DP$_C$(ϵ, δ;l,b,K): The query log anonymized by (ϵ, δ)-differentially private algorithm A_{Click} as presented in section 3.2.2, where l is the query click limits per user, b is the Laplacian noise scale, K is the frequency threshold. The output format of the log is the same as KA(K).
- DP$_S$(ϵ, δ;T_{Gap},l_s,l_q,b,K): This anonymized query log consists of the session-based differentially private output and a copy of DP$_C$(ϵ, δ;l,b,K). The session-based output query log is anonymized via $A_{Session}$ and A_{Click} as in section 3.2. T_{Gap} is the session timeout threshold in minutes, l_s is the session limits per user, l_q is the query limits per session, b is the Laplacian noise scale, K is the frequency threshold. The anonymized log contains both session and query clickthrough data.

5.2 Query Suggestion

The query suggestion approach we proposed earlier is based on the calculation of similarities between query pairs which can be generated from Q'. In this section, we use four different types of query logs as presented in section 5.1 to support query suggestion.

The effectiveness of a query suggestion approach is evaluated by comparing between the predicted ranked list of candidate queries and the ground truth. For each session in the test set, we perform query suggestion for each of the prefix query sequence and use the remaining queries of the session as the ground truth.

Table 5 presents Precision and Recall at the ranking position 5 for query suggestion, and the number of sessions in the test set that can still be used by query suggestion, the number of the evaluated or remaining sessions. The major parameters for the anonymized query logs we used are:

- KA(K=20)
- DP$_C$(ϵ=8,δ=2.25*10^{-7};l=4,b=1,K=20)
- DP$_S$(ϵ=8,δ=2.25*10^{-7};T_{Gap}=30, l_s=1,l_q=3,b=1,K=20).

where all three anonymized runs share the same frequency threshold value $k = 20$ for a fair comparison.

As we can see, the number of test sessions that our algorithm successfully evaluated in DP$_C$ and DP$_S$ (4,144 and 4,119) are much fewer than in KA (9,494) and Original (18,475). Such information loss is inevitable for getting strong privacy protection. The two runs based on differential privacy successfully suggest queries for a similar amount of sessions. The KA run based on k-anonymity suggests queries for sessions as twice many as the runs based on differential privacy would do. It is because k-anonymity doesn't limit the number of records from each individual while differential privacy does. Therefore, in terms of the quantity of the test sessions that could be evaluated, k-anonymity wins differential privacy, if both constrained by the same cut-off threshold k.

Table 5 also presents the IR utility measures: Precision and Recall. DP$_C$ and DP$_S$ outperform the other runs. Especially, DP_S achieves the best utility results in both Precision and Recall. The KA run works less effective in terms of IR utilities than DP$_C$ and DP$_S$.

An interesting finding is that the number of evaluated sessions contradicts with task-specific IR utility measures. That is, the runs releasing less sessions yield better IR utility scores. We think the underlying reason roots from the nature of IR. It is because (a) the records with the higher frequency (the more common ones) have a greater chance to be released by differential privacy; (b) they are also records that are positively correlated to producing relevant results for an IR task because they reflect similar behaviors from many different users and are better and more effective data records. That is to say, although DP$_C$ and DP$_S$ release fewer data and suggest less sessions/queries, their released content happen to be more useful to IR. This result is very encouraging for us to advocate the use of actual IR utility metrics over data percentage.

5.3 Parameter Settings

In Table 4, we compare the privacy level in DP$_S$ with different parameter settings. By showing the typical runs and their parameter settings, we observe that ϵ is very sensitive to the max number of sessions per user l_s and the max number of queries per session l_q. Moreover, δ is also very sensitive to the noise scale b and the frequency threshold K.

There are no hard rules for setting the parameters. Generally, smaller ϵ and δ values lead to stronger privacy guarantees but δ can not be too large. This gives the query log owner, usually the commercial search companies, more flexibility to pick proper privacy parameter values in order to achieve a good balance between privacy and utility. In Table 4, many listed runs are acceptable to use, for instance DP$_S$($\epsilon = 8, \delta = 2.25 \times 10^{-7}$; T_{Gap}=30, l_s=1,l_q=3,b=1,k=20) is one of the good runs that we use in the experiments.

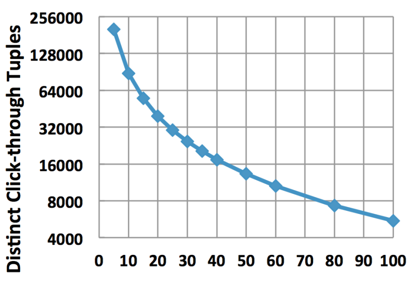
5.4 Privacy Utility Tradeoff

In this section, we run further experiments to analyze the privacy-utility tradeoff during query log anonymization. It is important to show the consequences of using varying anonymization algorithms and using different parameter settings. The comparisons and suggestions we provide in this section should be able to help data owners to make decisions when they need to anonymize a query log.

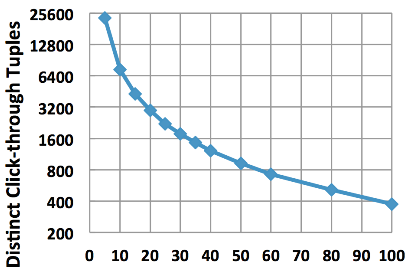
Figure 1 shows the scale of the anonymized data with varying frequency threshold K. Each data point in the figure corresponds to an anonymized query log. While changing the K values, we fix the other parameters in Figure 1 as T_{Gap}=30,l_s=1,l_q=3,$l = 4$, b=1. Figure 1 (a) presents the change of distinct clickthrough tuples in query logs anonymized by KA; Figure 1 (b) presents the change of distinct clickthrough tuples in query logs anonymized by DP$_C$, while Figure 1 (c) presents the variation of different released sessions by DP$_S$. According to Figure 1, the scale of the anonymized query log is very sensitive to the frequency threshold parameter K. The anonymized log suffers a significant amount of data loss as K increases. Based on Eq. 9, the differential privacy parameters ϵ and δ (especially δ) decrease as K increases, which leads to even stronger privacy. Hence, the stronger privacy we require for differential privacy, the more data would lose in the anonymized query logs.

Figure 2 shows the relations between the number of sessions being evaluated and (a) the frequency threshold K, (b) Precision@5 and (c) Recall@5 for the query suggestion task. Figure 2(a) reveals that if we want to evaluate a certain amount of sessions, the K used in differential privacy could be much less than the K used in k-anonymity. Figures 2(b) and (c) present the relationship between the application-based utility score and the amount of evaluated sessions. We observe that the differential privacy runs have better utility than the k-anonymity runs when the sessions being evaluated is no more than a certain value, around 5,500 in our case, while the k-anonymity runs may achieve even better utility than the differential privacy runs if there are more sessions evaluated.

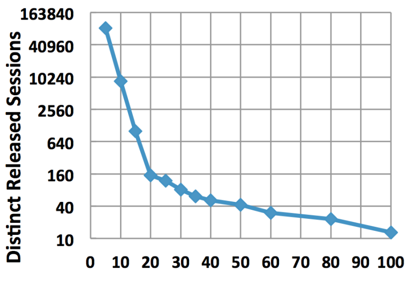
Moreover, we observe from the analysis and experimental results that fewer input records from each user lead to stronger privacy. According to the mathematical characteristics of (ϵ, δ)-differential privacy, the δ value is linearly related to the sensitivity value in our scenario. In other words, the fewer records we take from each user, the smaller δ value we can guarantee and thus achieves stronger privacy, and vice versa. For instance, if we double the number of input records per user l (or l_s), the anonymization mechanism will be with a doubled δ value. We, therefore, suggest to include raw data from more users while limiting fewer sessions and clickthroughs accepted from each user. The experiments based on different anonymization algorithms reveal an privacy-utility tradeoff. It seems that the balance between privacy and utility should be considered at the very beginning when we are selecting the parameters for the anonymization methods.



(a) KA Runs

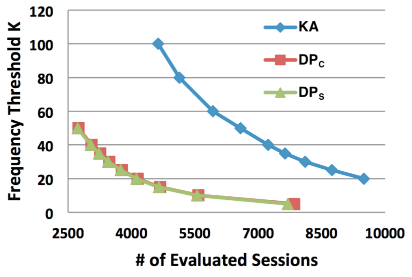


(b) DP$_C$ Runs

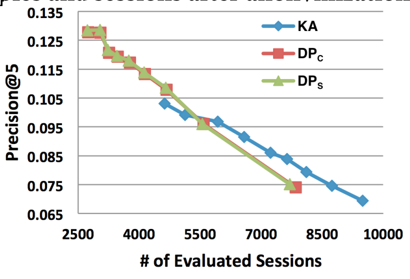


(c) DP$_S$ Runs

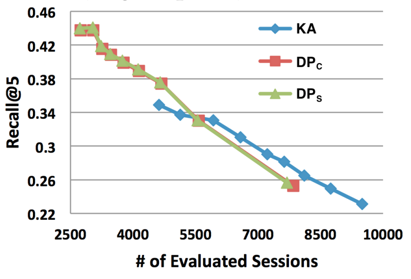
Figure 1: Data lost: distinct clickthrough tuples and sessions after anonymization with varying frequency threshold k values.



(a) Frequency Threshold K



(b) Precision@5



(c) Recall@5

Figure 2: Query Suggestion: Utility versus # of Evaluated Sessions

6 CONCLUSIONS

Query log anonymization is challenging. When a session of multiple queries are involved, it has becomes even more challenging. In this paper, we research on how to release session data from query logs with differential privacy. We propose methods to evaluate the utility of the anonymized session data and support query suggestion with those anonymized query logs. The results show that our session-based query log anonymization algorithm not only satisfies differential privacy but also is sufficiently capable of supporting complex IR applications. We wish this work can contribute to our research community to better enable data release for IR research.

7 ACKNOWLEDGMENTS

This research was supported by 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

[1] Eytan Adar. User 4XXXXX9: Anonymizing query logs. In *Query Logs Workshop at the WWW'07.*

[2] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *International Conference on Extending Database Technology.* Springer, 588–596.

[3] Michael Barbaro and Tom Zeller. Aug 2006. A Face Is Exposed for AOL Searcher No. 4417749. In *New York Times.*

[4] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. 2010. Discovering Frequent Patterns in Sensitive Data. In *KDD '10.*

[5] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query Suggestions in the Absence of Query Logs. In *SIGIR '11.*

[6] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query Suggestions Using Query-flow Graphs. In *WSCD '09.*

[7] Luca Bonomi and Li Xiong. 2013. A Two-phase Algorithm for Mining Sequential Patterns with Differential Privacy. In *CIKM '13.* ACM, New York, NY, USA.

[8] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware Query Suggestion by Mining Click-through and Session Data. In *KDD '08.*

[9] Claudio Carpineto and Giovanni Romano. Semantic Search Log K-anonymization with Generalized K-cores of Query Concept Graph. In *ECIR'13.*

[10] Rui Chen, Gergely Acs, and Claude Castelluccia. 2012. Differentially Private Sequential Data Publication via Variable-length N-grams. In *CCS '12.*

[11] Alissa Cooper. 2008. A Survey of Query Log Privacy-enhancing Techniques from a Policy Perspective. *ACM Trans. Web* 2, 4, Article 19 (Oct. 2008), 27 pages.

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

[13] Cynthia Dwork. 2011. *Differential Privacy.* Springer US, Boston, MA, 338–340.

[14] Michaela Gotz, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke. 2012. Publishing search logs:a comparative study of privacy guarantees. *IEEE Transactions on Knowledge and Data Engineering* 24, 3 (2012).

[15] Dongyi Guan, Sicong Zhang, and Hui Yang. Utilizing Query Change for Session Search. In *SIGIR '13.*

[16] Morgan Harvey, Claudia Hauff, and David Elsweiler. 2015. Learning by Example: Training Users with High-quality Query Suggestions. In *SIGIR '15.*

[17] Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, E. P. Lim, and H. Li. 2009. Web Query Recommendation via Sequential Query Prediction. In *ICDE'09.*

[18] Yuan Hong, Xiaoyun He, Jaideep Vaidya, Nabil Adam, and Vijayalakshmi Atluri. Effective Anonymization of Query Logs. In *CIKM '09.*

[19] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. "I Know What You Did Last Summer": Query Logs and User Privacy. In *CIKM '07.*

[20] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. Vanity Fair: Privacy in Querylog Bundles. In *CIKM '08.*

[21] Makoto P. Kato, Tetsuya Sakai, and Katsumi Tanaka. 2012. Structured Query Suggestion for Specialization and Parallel Movement: Effect on Search Behaviors. In *WWW '12.*

[22] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing Search Queries and Clicks Privately. In *WWW '09.*

[23] Haoran Li, Li Xiong, Xiaoqian Jiang, and Jinfei Liu. 2015. Differentially Private Histogram Publication for Dynamic Datasets: an Adaptive Sampling Approach. In *CIKM '15.*

[24] S. Su, S. Xu, X. Cheng, Z. Li, and F. Yang. 2015. Differentially Private Frequent Itemset Mining via Transaction Splitting. *IEEE TKDE* 27, 7 (July 2015).

[25] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[26] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. 2013. Differentially private histogram publication. *The VLDB Journal* 22, 6 (2013).

[27] S. Xu, S. Su, X. Cheng, Z. Li, and L. Xiong. 2015. Differentially private frequent sequence mining via sampling-based candidate pruning. In *ICDE'15.*

[28] Sicong Zhang, Grace Hui Yang, Lisa Singh, and Li Xiong. 2016. Safelog: Supporting Web Search and Mining by Differentially-Private Query Logs. In *2016 AAAI Fall Symposium Series.*

[29] Sicong Zhang, Hui Yang, and Lisa Singh. 2016. Anonymizing Query Logs by Differential Privacy. In *SIGIR '16.*