Releasing Search Queries and Clicks Privately

Aleksandra Korolova [*]
Computer Science Department
Stanford University
Stanford, CA
korolova@cs.stanford.edu

Krishnaram Kenthapadi, Nina Mishra, Alexandros Ntoulas
Search Labs
Microsoft Research
Mountain View, CA
{krisken, ninam, antoulas}@microsoft.com

ABSTRACT

The question of how to publish an anonymized search log was brought to the forefront by a well-intentioned, but privacy-unaware AOL search log release. Since then a series of ad-hoc techniques have been proposed in the literature, though none are known to be provably private. In this paper, we take a major step towards a solution: we show how queries, clicks and their associated perturbed counts can be published in a manner that rigorously preserves privacy. Our algorithm is decidedly simple to state, but non-trivial to analyze. On the opposite side of privacy is the question of whether the data we can safely publish is of any use. Our findings offer a glimmer of hope: we demonstrate that a non-negligible fraction of queries and clicks can indeed be safely published via a collection of experiments on a real search log. In addition, we select an application, keyword generation, and show that the keyword suggestions generated from the perturbed data resemble those generated from the original data.

Categories and Subject Descriptors: H.2.0 [Database Management]: Security, integrity, and protection; H.3 [Information Storage and Retrieval]; G.2.3 [Mathematics of Computing]; K.4 [Computers and Society];

General Terms: Algorithms, Experimentation, Human Factors, Legal Aspects, Measurement, Performance, Security, Theory

1. INTRODUCTION

Web search logs collect queries and clicks of users as they interact with a search engine. These logs have been successfully used by search engines in order to improve the quality of search results: for example, to fix spelling errors, suggest related searches, expand acronyms and estimate query popularity over time.

The release of these logs to the greater public would be a great benefit to society. Computer science researchers have been building a case for search log access [4, 25] so that they could study and analyze new IR algorithms via a common benchmark search log, as well as learn about user information needs and query formulation approaches. Social scientists could investigate the use of language in queries as well as discrepancies between user interests as revealed by their queries versus as revealed by face-to-face surveys [24]. Advertisers could use the logs to understand how users navigate to their pages, gain a better understanding of their competitors, and improve keyword advertising campaigns.

[*]Work done while interning at Microsoft Research.

On the other hand, the release of these logs to the greater public would be catastrophic from a privacy perspective. Users communicate with a search engine in an uninhibited manner, leaving behind an electronic trail of confidential thoughts and painfully identifiable information. For example, users search for their own name [17] or the names of their friends, home address, their health concerns, as well as names of their family and friends. Users even enter their credit card number and social security number just to find out what information is present on the web. It would be irresponsible for a search engine company to release this data without modification.

The open question to date is if there even exists a way to publish search logs in a perturbed fashion in a manner that is simultaneously useful and private. At first blush, the problem seems deceptively easy: why not just replace usernames with random identifiers? This simplistic view led to an AOL data release in 2006 in which the searches of an innocent citizen were quickly identified by a newspaper journalist [5]. As a consequence of releasing this private data set the CTO of AOL resigned, two employees were fired, a class action lawsuit is pending, and the event lives on as one of the "101 Dumbest Moments in Business" [14].

Further ad-hoc techniques have been explored in the literature. Kumar et al [19] consider tokenizing each search query and securely hashing the token into an identifier. Hashing gives only the appearance of privacy and, indeed, the authors show that hashes can be inverted by using token frequencies and other search logs.

An important lesson from decades of research in cryptography is that ad-hoc methods do not work. At issue is the fact that a data releaser does not know a priori what information an attacker will use in concert with the released data to deduce more private information. In the case of token-based hashing, prior search log releases were used. In the case of NetFlix [22], anonymized user ratings were combined with IMDb ratings to infer individuals.

1.1 Contributions

In this paper, we take a first significant step towards a solution. Rather than producing a search log, we consider the problem of releasing a query click graph. Rather than relying on intuition for showing that our approach provides privacy, we utilize a formal definition due to Dwork et al [11] that allows for an attacker with arbitrary prior knowledge, and design an algorithm for releasing a query-click graph that provably satisfies that definition.

Utility: We propose to publish a query click graph where the vertices correspond to both queries and URLs and there is an edge from a query to a URL with weight equal to the number of users who click on that URL given they posed the query. Each query node is labeled by the number of times this query was posed in the log. Similarly, there is an edge from one query to another query with weight equal to the number of users who posed one query and reformulated to another.

The query click graph is the basis for many uses of the search log [8, 3]. Query suggestions can be derived using common query reformulations. Spelling corrections can be inferred from queries with low click through and high reformulation rates. Similar queries can be found using common URLs clicked for those queries. Query classification and keyword generation can also be deduced from the query click graph [13].

Privacy: From the privacy side, we adapt the differential privacy definition [11]. In a nutshell, the definition states that upon seeing a published data set an attacker should gain little knowledge about any specific individual.

The algorithm for producing a Private Query Click graph is quite simple and can be intuitively described as "throw away tail queries":

Queries: To determine which queries to publish, if the frequency of the query plus some noise exceeds some threshold, we publish the query, otherwise we do not. We also publish the number of times the query was posed plus noise.

URLs: Given the queries that are safe to publish, the ten URLs surfaced are also safe to publish because anyone can pose a query to a search engine and see the top ten links. To publish the number of users who click on a result, we compute the actual number and add noise.

Our privacy proofs demonstrate that each of these steps individually preserves privacy and further that the composition of the steps also preserves privacy. In addition, we precisely characterize the trade-off between privacy and threshold used for publishing a query: the more stringent the privacy requirement, the higher the threshold, and consequently the fewer the number of queries that can be safely published.

Experiments: Given the privacy theorems, we next consider whether what we can publish from a real search log is of any use. We show that the fraction of distinct queries that can be published, as well as the amount of search volume involving those queries, is non-negligible. We then select two applications, keyword generation and studying human fears, and demonstrate that keywords and fears obtained from our perturbed logs closely resemble those obtained from the original unperturbed data. The experiments are an indication that it may be possible, if only in a limited sense, to bridge the gap between privacy and utility.

2. RELATED WORK

We describe work related to both search log anonymization and differential privacy.

2.1 Search Log Anonymization

The AOL search log release sparked interest in the problem of search log anonymization. In that data set, usernames were masked with random identifiers [2] and, in a matter of days, a New York Times reporter identified Thelma Arnold, a 62-year old widow from Lilburn, GA as user #4417749 [5], and her queries ranging from landscapers in her town to diseases of her friends.

Following AOL's release, many other ad-hoc techniques have been proposed. For example, if removing usernames is not enough, then it is natural to wonder if removing session IDs preserves privacy. It turns out that such an approach fails since one can design user models of approximate time that passes in between queries to stitch together sessions via the timestamps. Beyond inferring sessions, the heart of the problem lies in the fact that revealing a single query such as a credit card number breaks privacy.

If the queries themselves are private, then it is natural to wonder if hashing the queries preserves privacy. In fact, that too fails as Kumar et al [19] nicely argue. They show that tokenizing a query, hashing the tokens and publishing the hashes does not preserve privacy since an adversary who has access to another log can reverse-engineer the tokens by utilizing the frequency with which the query appears in the log.

Jones et al [16] study an application of simple classifiers to connect a sequence of queries to the location, gender and age of the user issuing the queries, and argue that releasing a query log poses a privacy threat because these three characteristics of the user can be used to create a set of candidate users who might have posed that query. Their more recent work [17] investigates privacy leaks that are possible even when queries from multiple users are grouped together and no user or session identifiers are released.

In short, while many papers describe ad-hoc techniques [25, 1] the results are, by and large, negative for privacy. Thus, the question of how to anonymize a search log remains open.

Our work focuses on a seemingly more attainable goal of releasing a private query click graph. While this graph is not as powerful as the actual search log, many computations can still be performed on the click graph with results similar to the actual search log, e.g., finding similar queries, keyword generation, and performing spell corrections.

2.2 Differential Privacy

If there is any lesson to be drawn from decades of cryptography and the many privacy compromises in published datasets, it is that ad-hoc techniques do not work [9]. The way to achieve privacy is to start with a rigorous definition and design an algorithm that satisfies the privacy definition.

Our paper is the first to take a concrete definition of privacy, differential privacy, and design an algorithm for producing a private query click graph that provably satisfies that definition. We choose differential privacy because it does not stipulate the prior knowledge of the attacker: regardless of what the attacker knows, releasing data that satisfies the differential privacy definition will not substantially increase the attacker's chance of inferring something about an individual. A formal definition appears in Section 3.

Many algorithms exist for publishing data in a differentially private manner. The seminal work of Dwork et al [11] shows that any function with low sensitivity can be computed privately. A function has *low sensitivity* if the addition or removal of one person can only change the outcome of the function evaluation by a small amount. We use the ideas developed in [11] for our analysis. The aggregate statistical values reflected in our query click graph have low sensitivity, provided that each user issues a bounded number of queries.

Randomly sampling a data set is known to be differentially private [7], but only under the assumption that there are few rare values. In the web context, it is more likely that every person's searches are a unique fingerprint to them; thus, randomly sampling the search log breaks privacy.

Prior to our work, no differentially private algorithm was known for efficiently publishing a query click graph. A mechanism due to McSherry and Talwar [20] and Blum et al [6] could be adapted to this task in theory but is not feasible in practice, as this mechanism takes time exponential in the size of the output space. More recently, McSherry and Talwar have proposed an algorithm for releasing synthetic queries [21] that holds promise for synthetic data releases.

3. PRIVACY DEFINITION

In this section we describe the differential privacy definition that was first conceived by Dwork et al [11]. Intuitively, a data releaser preserves privacy if no attacker gains significant knowledge about an individual's private searches beyond what the attacker could have learned from a similar neighboring search log in which that individual's searches are modified or removed. To formalize this intuition, we require that for all pairs of neighboring search logs that differ in one user's searches and all possible published query click graphs, the probability that any subset of query click graphs is published from one log is within an e^ϵ multiplicative factor of the probability that that subset is published from the neighboring log, plus an additive δ component.

Definition 1. A randomized algorithm A is (ϵ, δ)*-differentially private* if for all data sets D_1 and D_2 differing in at most one user and all $\hat{D} \subseteq Range(A)$: $\Pr[A(D_1) \in \hat{D}] \leq e^\epsilon \cdot \Pr[A(D_2) \in \hat{D}] + \delta$.

Differential privacy captures the desired notion of privacy that the risk to one's privacy should not substantially increase as a result of participating in the dataset (e.g. as a result of using the search engine), by guaranteeing that any given disclosure is almost as likely whether or not the individual participates in the dataset. Differential privacy also does not make any assumptions about adversary's computational power or ability to access additional data beyond the release. Although differential privacy is not an absolute privacy guarantee, in the setting when we have to trade-off the benefits of the data release with user privacy, differential privacy ensures that the amount of additional privacy risks incurred by users is limited.

In the first definition of differential privacy [11], $\delta = 0$. Subsequent work [10] relaxed the definition to include a non-zero additive component.

There are no hard and fast rules for setting ϵ and δ – it is generally left to the data releaser. One consideration to take into account when choosing δ is the number of users participating in the dataset. Indeed, imagine that every person's every query is private - e.g. the log consists of searches for names and social security numbers. Then $\delta > \frac{1}{\text{number of users}}$ suggests that at least one person's privacy is compromised. Of course, imagining that a search log consists of only sensitive queries is a very paranoid view of privacy but nonetheless, the magnitude of $\frac{1}{\text{number of users}}$ is useful to keep in mind when choosing δ.

4. ALGORITHM FOR RELEASING SEARCH QUERIES AND CLICKS

We next describe our algorithm for generating a private query click graph. The key components are: determining which queries to publish, together with the number of times the query was posed and, further, determining which URLs to publish, together with the number of times the URL was clicked for each query. Our basic method for accomplishing these tasks utilizes a noisy count: for any statistic x of the data, the *noisy count* of x is $x+$ Lap(b), where Lap(b) denotes a random variable drawn independently from the Laplace distribution with mean zero and scale parameter b.

At a high-level, the algorithm proceeds as follows.

1. **Limit User Activity:** Keep only the first d queries posed by each user and first d_c URL clicks of each user (Line 2 of Algorithm 1).

2. **Queries:** If the noisy count of the number of times a query is posed exceeds a specified threshold, output the query together with its noisy count (Lines 3-5 of Algorithm 1).

3. **URLs:** If a query is safe to publish, then the ten URLs that are surfaced for that query are also safe to publish since anyone can pose the query to a search engine and see the ten results. For each query and ten surfaced URLs for that query, we output the noisy count of the number of times each URL was clicked for that query (Line 6 of Algorithm 1).

Some comments about the algorithm are in order.
(1) We limit user activity in order to preserve privacy: if a user can contribute an unbounded number of queries and clicks then they can have an unlimited influence on the set of queries that are published (and the URLs that are clicked). In practice, a typical user does not pose an unlimited number of queries anyway — studies suggest that an average user poses about 34 queries per month [12].
(2) While the main idea of throwing away tail queries is quite natural and has been previously suggested in the literature [1], this paper is the first to mathematically quantify exactly how to perform this operation so as to preserve privacy with respect to a rigorous privacy definition. Indeed, our theorems in subsequent sections establish a direct connection between the threshold used for defining a tail query and the resulting privacy guarantees.
(3) Because our algorithm does not release any fake queries (i.e. the queries it releases are a subset of the queries present in the original log), it won't satisfy differential privacy with $\delta = 0$. For example, in the event that a user poses a search query q in log D_1 that does not appear in the neighboring log D_2, then there's a non-zero probability that q is published when starting from log D_1 and a zero probability that q is published when starting from log D_2. Hence the former probability cannot be upper bounded by the latter probability without the extra help of a non-zero additive component δ.
(4) It is crucial to note that every time our algorithm computes the noisy count, the noise should be generated independently from the Laplace distribution.

Algorithm 1 Release-Data

```
1: Input: $D$ - search log, $d, d_c$ - parameters that limit user activity, $b, b_q, b_c$ - noise parameters, $K$ - threshold that defines tail.
2: Limit-User-Activity: $D \leftarrow$ Keep the first $d$ queries and the first $d_c$ URL clicks of each user.
3: For each query $q$, let $M(q, D) =$ number of times $q$ appears in $D$
4: Select-Queries: $Q \leftarrow \{q : M(q, D) + Lap(b) > K\}$
5: Get-Query-Counts: For each $q$ in $Q$, output $\langle q, M(q, D) + Lap(b_q)\rangle$
6: Get-Click-Counts: For each URL $u$ in the top ten results for $q \in Q$, output $\langle q, u,$ number of times $u$ was clicked when $q$ was posed $+Lap(b_c)\rangle$.
```

5. PRIVACY GUARANTEES

We now state formally the (ϵ, δ)-differential privacy guarantees that our algorithm provides. We then provide a sketch of the proof that each of the individual steps preserves privacy and, further, that their composition preserves privacy.

Let K, d, d_c, b, b_q, b_c be the parameters of Algorithm 1 such that $K \geq d$. Define $\alpha = \max(e^{1/b}, 1 + \frac{1}{2e^{(K-1)/b} - 1})$ and the multiplicative and additive privacy parameters as $\epsilon_{alg} = d \cdot \ln(\alpha) + d/b_q + d_c/b_c$ and $\delta_{alg} = \frac{d}{2}\exp(\frac{d-K}{b})$.

THEOREM 1. *Algorithm 1 is $(\epsilon_{alg}, \delta_{alg})$-differentially private for every pair of search logs differing in one user, where ϵ_{alg} and δ_{alg} are defined as above.*

5.1 Proof Overview

In order to prove Theorem 1, we will show that each step of the algorithm is differentially private for appropriate values of ϵ and δ and that their composition is also differentially private.

Lemma 1. **Select-Queries** is $(d \cdot \ln(\alpha), \delta_{alg})$-differentially private if each user is limited to posing at most d queries, where α and δ_{alg} are defined as above.

Lemma 2. **Get-Query-Counts** is $(d/b_q, 0)$-differentially private.

Lemma 3. **Get-Click-Counts** is $(d_c/b_c, 0)$-differentially private.

Lemma 4. Suppose **Select-Queries** is (ϵ_1, δ)-differentially private, **Get-Query-Counts** is $(\epsilon_2, 0)$-differentially private, and **Get-Click-Counts** is $(\epsilon_3, 0)$-differentially private. Then Algorithm 1 is $(\epsilon_1 + \epsilon_2 + \epsilon_3, \delta)$-differentially private.

Theorem 1 follows from Lemmas 1, 2, 3, and 4. We next sketch the key ideas in the proofs of the above lemmas.

5.2 Privacy for Selecting Queries

We first prove the guarantees for **Select-Queries** when each user can pose at most one query (Lemma 5) and then generalize the proof to hold for d queries per user to obtain Lemma 1.

Privacy for Select-Queries with $d = 1$:
Let $\alpha = \max(e^{1/b}, 1 + \frac{1}{2e^{(K-1)/b}-1})$ and $\delta_1 = \frac{1}{2}e^{\frac{1-K}{b}}$. Denote the Select-Queries algorithm by A.

Lemma 5. If each user can pose at most one query and $K \geq 1$, then Select-Queries satisfies $(\ln(\alpha), \delta_1)$-differential privacy.

PROOF. Let D_1 and D_2 be arbitrary search logs that differ in exactly one query q_* posed by some user such that D_2 is the larger of the two logs. Let $\hat{D} \subseteq Range(A)$ denote an arbitrary set of possible outputs. Then we need to show the following:

$$Pr[A(D_1) \in \hat{D}] \leq \alpha Pr[A(D_2) \in \hat{D}] + \delta_1 \quad (1)$$

$$Pr[A(D_2) \in \hat{D}] \leq \alpha Pr[A(D_1) \in \hat{D}] + \delta_1 \quad (2)$$

We consider two different scenarios depending on whether q_* already appears as a query of some user in D_1 or it is a new query. For the first case, we do not need the additive parameter (i.e., $\delta_1 = 0$ works) as we can bound the ratio of the probabilities in each expression above. The key idea is to notice that q_* has a slightly higher probability of being released from D_2 compared to D_1 and to argue that the ratio of these probabilities can be bounded. However, for the second case, q_* can never be released from D_1, and hence we cannot bound the ratio of the probabilities. Instead, we make use of the fact that q_* occurs only once in D_2 and use the additive parameter δ_1 to bound the probability that its noisy count exceeds the threshold K.

Recall that our algorithm A only produces a subset of queries contained in D_1 (or D_2). Hence, any set of queries O that contains some query not present in D_1 or D_2 need not be considered as part of $\hat{D}$ in our analysis, as the probability that A produces O is zero, with D_1 or D_2 as input. We also partition $\hat{D}$ into two subsets: $\hat{D}^+$, the query sets in $\hat{D}$ that contain q_* and $\hat{D}^-$, the query sets in $\hat{D}$ that do not contain q_*.

Throughout the proof, we will utilize Observations 2, 3 and 4 described in the Appendix, which are useful tools for dealing with properties of ratios and of the Laplace distribution.

Case 1: $q_* \in D_1$
Let $M(q, D)$ be the number of times query q appears in the search log D. Then, $M(q_*, D_1) \geq 1, M(q_*, D_2) = M(q_*, D_1) + 1$.

We first prove inequality (1) with $\alpha = e^{1/b}$ and $\delta_1 = 0$ by upper bounding the ratio $\frac{Pr[A(D_1)\in\hat{D}]}{Pr[A(D_2)\in\hat{D}]}$, which we denote by R_2^1.

From our partition of $\hat{D}$ into $\hat{D}^+$ and $\hat{D}^-$ and using observation 4,[1] we have:
$R_2^1 = \frac{Pr[A(D_1)\in\hat{D}]}{Pr[A(D_2)\in\hat{D}]} = \frac{Pr[A(D_1)\in\hat{D}^+]+Pr[A(D_1)\in\hat{D}^-]}{Pr[A(D_2)\in\hat{D}^+]+Pr[A(D_2)\in\hat{D}^-]}$
$\leq \max\left(\frac{Pr[A(D_1)\in\hat{D}^+]}{Pr[A(D_2)\in\hat{D}^+]}, \frac{Pr[A(D_1)\in\hat{D}^-]}{Pr[A(D_2)\in\hat{D}^-]}\right)$

Consider now the ratio $\frac{Pr[A(D_1)\in\hat{D}^+]}{Pr[A(D_2)\in\hat{D}^+]}$. Recall that in our algorithm, the decision to release a particular query is made independently for each query and that D_1 and D_2 differ only in the number of times that q_* occurs in each of them. Hence, for a particular possible output O, s.t. $q_* \in O$: $\frac{Pr[A(D_1)=O]}{Pr[A(D_2)=O]} = \frac{Pr[q_* \text{released by } A(D_1)]}{Pr[q_* \text{released by } A(D_2)]}$.

Generalizing this observation to all outputs $O_i \in \hat{D}^+$ we obtain:
$\frac{Pr[A(D_1)\in\hat{D}^+]}{Pr[A(D_2)\in\hat{D}^+]} = \frac{\sum_{O\in\hat{D}^+} Pr[A(D_1)=O]}{\sum_{O\in\hat{D}^+} Pr[A(D_2)=O]} = \frac{Pr[q_* \text{released by } A(D_1)]}{Pr[q_* \text{released by } A(D_2)]} =$
$\frac{Pr[M(q_*,D_1)+Lap(b)>K]}{Pr[M(q_*,D_2)+Lap(b)>K]} = \frac{Pr[M(q_*,D_1)+Lap(b)>K]}{Pr[M(q_*,D_1)+1+Lap(b)>K]}$

By analogous reasoning with respect to $\hat{D}^-$ we obtain:
$\frac{Pr[A(D_1)\in\hat{D}^-]}{Pr[A(D_2)\in\hat{D}^-]} = \frac{Pr[M(q_*,D_1)+Lap(b)<K]}{Pr[M(q_*,D_2)+Lap(b)<K]} = \frac{Pr[M(q_*,D_1)+Lap(b)<K]}{Pr[M(q_*,D_1)+1+Lap(b)<K]}$

From these two bounds on the ratios, we bound R_2^1:
$R_2^1 \leq \max\left(\frac{Pr[M(q_*,D_1)+Lap(b)>K]}{Pr[M(q_*,D_1)+1+Lap(b)>K]}, \frac{Pr[M(q_*,D_1)+Lap(b)<K]}{Pr[M(q_*,D_1)+1+Lap(b)<K]}\right)$,
which by Observation 3 implies that

$$R_2^1 = \frac{Pr[A(D_1) \in \hat{D}]}{Pr[A(D_2) \in \hat{D}]} \leq \max(1, e^{1/b}) = e^{1/b} \quad (3)$$

proving inequality (1), as desired.

A similar analysis can be performed to show that

$$\frac{Pr[A(D_2) \in \hat{D}]}{Pr[A(D_1) \in \hat{D}]} \leq e^{1/b}, \quad (4)$$

yielding the proof of inequality (2) with $\alpha = e^{1/b}$ and $\delta_1 = 0$.

Case 2: $q_* \notin D_1, q_* \in D_2$
We now proceed to prove inequality (1) with $\alpha = \frac{1}{1-0.5\exp(\frac{1-K}{b})}$ and $\delta_1 = 0$.

First consider outputs that do not contain the new query q_*. By similar reasoning as in Case 1, the probability of obtaining an output O, where $O \in \hat{D}^-$ when starting from the D_2 log differs from the probability of obtaining an output O, when starting from D_1 only in the choice that the algorithm has to make for query q_*. Hence, $Pr[A(D_2) \in \hat{D}^-] = Pr[q_* \text{ was not released by} A(D_2)] \cdot Pr[A(D_1) \in \hat{D}^-]$ and

$$\frac{Pr[A(D_1) \in \hat{D}^-]}{Pr[A(D_2) \in \hat{D}^-]} = \frac{1}{Pr[q_* \notin A(D_2)]} \quad (5)$$

$$= \frac{1}{Pr[1 + Lap(b) < K]} = \frac{1}{1 - 0.5\exp(\frac{1-K}{b})} \quad (6)$$

Since query q_* is not present in D_1, our algorithm would not produce any output containing q_* when given D_1 as input, and so $Pr[A(D_1) \in \hat{D}^+] = 0$.

Using the partition of $\hat{D}$ into $\hat{D}^+$ and $\hat{D}^-$ we have:

$$\frac{Pr[A(D_1) \in \hat{D}]}{Pr[A(D_2) \in \hat{D}]} = \frac{Pr[A(D_1) \in \hat{D}^-]}{Pr[A(D_2) \in \hat{D}^+] + Pr[A(D_2) \in \hat{D}^-]}$$
$$\leq \frac{Pr[A(D_1) \in \hat{D}^-]}{Pr[A(D_2) \in \hat{D}^-]} \leq \frac{1}{1 - 0.5\exp(\frac{1-K}{b})} \quad (7)$$

proving inequality (1), as desired.

[1]Observation 4 holds only for positive denominators. In the event that either or both the denominators are zero, it can be shown that the differential privacy bound continues to hold.

It remains to show that in this case, inequality (2) is satisfied with $\alpha = 1 - 0.5\exp(\frac{1-K}{b})$ and $\delta_1 = 0.5\exp(\frac{1-K}{b})$.

Observe that

$$Pr[A(D_2) \in \hat{D}^+] \le Pr[q_* \text{ was released}] =$$
$$= Pr[M(q_*, D_2) + Lap(b_1) > K] = 0.5\exp(\frac{1-K}{b}) \quad (8)$$

Combining (8) and (6), we have:
$\frac{Pr[A(D_2)\in\hat{D}]}{Pr[A(D_1)\in\hat{D}]} = \frac{Pr[A(D_2)\in\hat{D}^+]+Pr[A(D_2)\in\hat{D}^-]}{Pr[A(D_1)\in\hat{D}^-]} = \frac{Pr[A(D_2)\in\hat{D}^-]}{Pr[A(D_1)\in\hat{D}^-]} + \frac{Pr[A(D_2)\in\hat{D}^+]}{Pr[A(D_1)\in\hat{D}^-]} \le 1 - 0.5\exp(\frac{1-K}{b}) + \frac{0.5\exp(\frac{1-K}{b})}{Pr[A(D_1)\in\hat{D}]}$,
which completes the proof of inequality (2).

Thus from the two cases, depending on whether q_* is an additional occurrence of an element already present in D_1 or it is an entirely new element to D_1, we established that our algorithm satisfies the $(\ln(\alpha), \delta_1)$-differential privacy, where
$\alpha = \max\left(e^{1/b}, 1 - 0.5\exp(\frac{1-K}{b}), \frac{1}{1-0.5\exp(\frac{1-K}{b})}\right) =$
$= \max(e^{1/b}, \frac{1}{1-0.5\exp(\frac{1-K}{b})})$, and $\delta_1 = \frac{1}{2}e^{(\frac{1-K}{b})}$. □

Observation 1. **[Need for δ_1]** A crucial observation made in the proof of this lemma is that the necessity for δ_1 arises only when the extra query in D_2 is a query that was not previously present in D_1, and we are attempting to upper bound the ratio of $\frac{Pr[A(D_2)\in\hat{D}]}{Pr[A(D_1)\in\hat{D}]}$.

We will use this observation next as we generalize the proof of privacy guarantees to the case where each user can pose at most d queries.

Privacy for Select-Queries for arbitrary d:
We next prove Lemma 1. We first show that straight-forward generalization does not work and hence perform a tighter analysis.

Straight-forward generalization: A natural approach towards this proof is to observe that a search log D_2 that differs from D_1 by at most d queries can be obtained from D_1 by adding the d queries to it, one query at a time. This enables the repeated application of the results of Lemma 5 to obtain:
$Pr[A(D_1) \in \hat{D}] \le \alpha Pr[A(D_1 + q_1) \in \hat{D}] + \delta_1$
$\le \alpha(\alpha Pr[A(D_1 + q_1 + q_2) \in \hat{D}] + \delta_1) + \delta_1 \le \cdots$
$\le \alpha^d Pr[A(D_2) \in \hat{D}] + \delta_1 \frac{\alpha^d - 1}{\alpha - 1}$

However, this approach yields $\delta_{alg} = \delta_1 \frac{\alpha^d-1}{\alpha-1}$, which will quickly exceed 1, yielding meaningless privacy guarantees. To avoid the blow-up in the additive component of privacy guarantees, we build on the insights of the Proof of Lemma 5, and especially, on Observation 1 in order to show better guarantees for δ_{alg}.

Tighter analysis for δ_{alg}:
As in the proof of Lemma 5, let D_2 be the larger of the two search logs, containing an additional d queries compared to D_1. Denote by $x_1, \ldots, x_{n_x}$ those of the additional d queries that are already in D_1 and by $y_1, \ldots, y_{n_y}$ - those queries that are unique to D_2. Note that
$\sum_{i=1}^{n_x}(M(x_i, D_2) - M(x_i, D_1)) + \sum_{i=1}^{n_y} M(y_i, D_2) \le d$ and $n_x + n_y \le d$

We also split the elements of $\hat{D}$ into two subsets as follows: denote by $\hat{D}^-$ the set of elements of $\hat{D}$ which can be obtained from both D_1 and D_2, and by $\hat{D}^+$ - the set of elements of $\hat{D}$ which can only be obtained from D_2 (as before, we remove those elements of $\hat{D}$ that cannot be obtained from either D_1 or D_2 from consideration wlog).

Observe that in the proof of Lemma 5, the additive component δ arose only when considering the ratio $\frac{Pr[A(D_2)\in\hat{D}]}{Pr[A(D_1)\in\hat{D}]}$ and not when considering the ratio $\frac{Pr[A(D_1)\in\hat{D}]}{Pr[A(D_2)\in\hat{D}]}$. We take advantage of this observation by proving the necessary upper bounds by recursively applying Lemma 5 in one case, and by performing a more careful analysis for the need for the additive component in the other case.

Proof of (1) with $\alpha_d = \alpha^d, \delta_{alg} = 0$:
We first observe that $\frac{Pr[A(D_1)\in\hat{D}]}{Pr[A(D_2)\in\hat{D}]}$ can be represented as a product of ratios of obtaining an output in $\hat{D}^-$ when starting from datasets differing in one element as follows:
$\frac{Pr[A(D_1)\in\hat{D}]}{Pr[A(D_2)\in\hat{D}]} = \frac{Pr[A(D_1)\in\hat{D}^-]}{Pr[A(D_2)\in\hat{D}^+]+Pr[A(D_2)\in\hat{D}^-]} \le \frac{Pr[A(D_1)\in\hat{D}^-]}{Pr[A(D_2)\in\hat{D}^-]} = \frac{Pr[A(D_1)\in\hat{D}^-]}{Pr[A(D_1+x_1)\in\hat{D}^-]} \cdot \frac{Pr[A(D_1+x_1)\in\hat{D}^-]}{Pr[A(D_1+x_1+x_2)\in\hat{D}^-]} \cdot \ldots \cdot \frac{Pr[A(D_1+x_1+\ldots x_{n_x}+y_1+\ldots+y_{n_y-1})\in\hat{D}^-]}{Pr[A(D_1+x_1+\ldots x_{n_x}+y_1+\ldots+y_{n_y})\in\hat{D}^-]}$

Applying the above decomposition of the ratio into a product of ratios [2] and the results of intermediate steps (3) and (6) of Lemma 5 to each of the ratios in the product, we obtain: $\frac{Pr[A(D_1)\in\hat{D}]}{Pr[A(D_2)\in\hat{D}]} \le \alpha^d$, as desired.

Proof of (2) with $\alpha_d = \alpha^d, \delta_{alg} = \frac{d}{2}\exp(\frac{d-K}{b})$:
By definition of $\hat{D}^+$ one can obtain an output in $\hat{D}^+$ when given the search log D_2, only if at least one of the queries in D_2 which was not present in D_1 is chosen for release. Applying the union bound we have:
$Pr[A(D_2) \in \hat{D}^+] \le \sum_{i=1}^{n_y} Pr[n_y \text{ was chosen for release}] = \sum_{i=1}^{n_y} Pr[M(y_i, D_2) + Lap(b) > K] \le$
$\sum_{i=1}^{n_y} Pr[Lap(b) > K - M(y_i, D_2)] = \frac{1}{2}\sum_{i=1}^{n_y}\exp\left(\frac{M(y_i,D_2)-K}{b}\right)$
(applying the knowledge that $n_y \le d$ and $M(y_i, D_2) \le d$)
$\le \frac{d}{2}\exp\left(\frac{d-K}{b}\right)$ (9)

Decomposing the ratio of outputs being in $\hat{D}^-$ as a product[3] of obtaining $\hat{D}^-$ when starting with dataset differing in one element we have:
$\frac{Pr[A(D_2)\in\hat{D}^-]}{Pr[A(D_1)\in\hat{D}^-]} = \frac{Pr[A(D_1+x_1)\in\hat{D}^-]}{Pr[A(D_1)\in\hat{D}^-]} \cdot \frac{Pr[A(D_1+x_1+x_2)\in\hat{D}^-]}{Pr[A(D_1+x_1)\in\hat{D}^-]} \cdot \ldots \cdot \frac{Pr[A(D_1+x_1+\ldots x_{n_x}+y_1+\ldots+y_{n_y})\in\hat{D}^-]}{Pr[A(D_1+x_1+\ldots x_{n_x}+y_1+\ldots+y_{n_y-1})\in\hat{D}^-]} \le$
(applying the result of (4) to those datasets differing in a query already present and the result of (5) to those datasets differing in a query not yet present)
$\le \prod_{i=1}^{n_x} e^{1/b} \cdot \prod_{i=1}^{n_y}\max(e^{1/b}, Pr[M(y_i, D_2) + Lap(b) < K])$
(using the value of $M(y_i, D_2)$ maximizing the probability term for each i) $\le e^{n_x/b} \cdot \left(\max(e^{1/b}, Pr[1 + Lap(b) < K])\right)^{n_y} =$
$= e^{n_x/b} \cdot \left(\max(e^{1/b}, 1 - 0.5\exp(\frac{1-K}{b}))\right)^{n_y} \le e^{\frac{n_x+n_y}{b}} \le \alpha^d$

We now use (9) and the inequality above to obtain the desired upper bound:
$\frac{Pr[A(D_2)\in\hat{D}]}{Pr[A(D_1)\in\hat{D}]} = \frac{Pr[A(D_2)\in\hat{D}^-]+Pr[A(D_2)\in\hat{D}^+]}{Pr[A(D_1)\in\hat{D}^-]} = \frac{Pr[A(D_2)\in\hat{D}^-]}{Pr[A(D_1)\in\hat{D}^-]} + \frac{Pr[A(D_2)\in\hat{D}^+]}{Pr[A(D_1)\in\hat{D}^-]} \le \alpha^d + \frac{0.5d\exp(\frac{d-K}{b})}{Pr[A(D_1)\in\hat{D}^-]} = \alpha^d + \frac{0.5d\exp(\frac{d-K}{b})}{Pr[A(D_1)\in\hat{D}]}$,
hence $Pr[A(D_2) \in \hat{D}] \le \alpha^d Pr[A(D_1) \in \hat{D}] + 0.5d\exp(\frac{d-K}{b})$, as desired.

[2] As long as $Pr[A(D_1) \in \hat{D}^-] \neq 0$, the denominator of all the ratios involved in the product is guaranteed to be non-zero; whereas, if $Pr[A(D_1) \in \hat{D}^-] = 0$, then the upper bound of α^d holds for it automatically.

[3] The denominator of any of the product terms is 0 only if $Pr[A(D_1) \in \hat{D}^-] = 0$, in which case the desired differential privacy guarantees follow from (9) .

5.3 Privacy for Noisy Counts

We next show that the steps involving noisy counts (**Get-Query-Counts** and **Get-Click-Counts**) are differentially private. We reduce both these steps to the problem of releasing histograms privately [11].

Private Release of Histograms: Consider an arbitrary domain X which has been partitioned into r disjoint bins. A histogram function, $f : X^n \rightarrow \mathcal{Z}^r$ maps the database points into these bins and reports the number of points in each bin. The sensitivity $S(f)$ of a function f denotes the maximum "change" in the value of f when the inputs differ in a single entry, i.e., $S(f) = max\{\|f(\mathbf{x}) - f(\mathbf{x'})\|_1 : x, x' \in X^n$ differ in a single entry $\}$. We use the following result:

THEOREM 2. *[11] For all $f : X^n \rightarrow R^r$ the following mechanism satisfies ϵ-differential privacy: $San_f(\boldsymbol{x}) = f(\boldsymbol{x}) + (Y_1, \ldots, Y_d)$, where the Y_i are drawn i.i.d. from $Lap(S(f)/\epsilon)$. Note that the privacy guarantees do not depend on r.*

Intuitively, this result shows that the amount of noise to be added to preserve privacy is closely related to the amount that any single argument to f can change its output.

For **Get-Query-Counts**, the reduction is as follows: the domain X (the search log) is partitioned into bins (distinct queries) according to Q. Function f reports the number of elements in the bin for each bin, i.e., the number of occurrences of each query. The datasets differ in one user, who can pose at most d queries, hence $S(f) = d$. Therefore, by Theorem 2 adding $Lap(d/\epsilon)$ noise to each query occurrence count gives ϵ-differential privacy guarantee, and **Get-Query-Counts** is d/b_q-differentially private.

Similarly for **Get-Click-Counts**, each query-URL pair serves as a partition bin (note that the top 10 URLs returned upon searching for a given query are not private and hence the partition is known given the set of queries Q). Hence, by Theorem 2, adding $Lap(b_c)$ noise to the true count of the number of clicks on a URL for a query will preserve d_c/b_c-differential privacy.

5.4 Privacy for Composition of Individual Steps

We have shown that steps 4–6 of the **Release-Data** algorithm preserve privacy, so it remains to show that limiting the user activity and the composition of the steps preserves privacy and to quantify the effect of applying these algorithms in sequence on the privacy guarantees (Lemma 4).

We note that **Limit-User-Activity** helps to satisfy the condition in Lemma 1 that each user should pose at most d queries and otherwise does not affect the privacy guarantees. To prove Lemma 4, we note that a straight-forward composition does not work, as it would cause the additive privacy parameter to blow up. We use a more careful analysis, which we omit due to lack of space.

6. DISCUSSION

The algorithm and analysis leaves open much room for discussion. How does a data releaser set the various parameters in the analysis above? Is the algorithm really different from just publishing queries with sufficiently large frequency (without the addition of noise)? What frequency queries end up getting published? Why do we recompute noise in Step 5? What happens if a count is negative after adding noise? Is it possible to release query reformulations? In this section we provide answers to each of these questions in turn.

Setting parameters. Given the numerous parameters in the theorems just proved, a natural question is how to set them. As mentioned earlier, it is up to the data releaser to choose ϵ, while it is advisable that $\delta < 1/n$, where n is the number of users. What about the remaining parameters? Lemma 1 offers answers for optimally setting[4] the threshold K and noise b when the desired privacy parameters and the limit for the number of queries per user are known: $K = d\left(1 - \frac{\ln(\frac{2\delta}{d})}{\epsilon}\right)$ and $b = \frac{d}{\epsilon}$.

d	1	5	10	20	40	80	160
K	5.70	31.99	66.99	140.00	292.04	608.16	1264.49
b	0.43	2.17	4.34	8.69	17.37	34.74	69.49

Table 1: Optimal choices of the threshold, K and noise, b as a function of d for fixed privacy parameters, $e^\epsilon = 10, \delta = 10^{-5}$

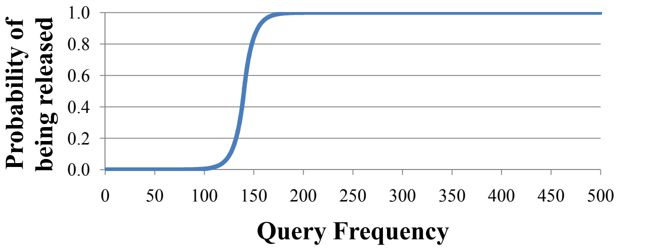

Figure 1: Probability of a query being released as a function of its frequency, for $d = 20, K = 140$, and $b = 8.69$

Table 1 shows how the optimal choices of threshold K and noise b vary as a function of the number of queries allowed per user, d, for fixed privacy parameters, $e^\epsilon = 10$ and $\delta = 10^{-5}$.

Publishing head queries. An important and natural question is why not just publish the queries with frequency larger than an intuitively-pleasing large number? The answer is that any such deterministic algorithm is provably not differentially private [23]. Beyond that, how should one even select such a large number? Our approach has the advantage that the threshold value K can be determined purely from the privacy parameters, ϵ and δ, and d. The values K and b are independent of the characteristics of the search log.

Which queries are published. If the algorithm is different than publishing the queries with sufficiently high frequency, it is reasonable to wonder which frequency queries do get published? Consider Figure 1 which shows the probability of the query being chosen for release as a function of its frequency, for $d = 20, K = 140$, and $b = 8.69$. The queries whose frequency is above 170 are virtually guaranteed to be released, the queries whose frequency is below 110 are virtually guaranteed not to be released, and the queries whose frequency is between 110 and 170 might or might not be released depending on the random choices of the algorithm. It is clear that since b is smaller than K, **Select-Queries** is close to the intuitive sharp threshold algorithm of only releasing the "head queries", where the "head queries" are defined as those whose frequency is above 170.

Fresh noise. Next we explain why we recompute the noisy query frequency count in Step 5 of **Release-Data**. Technically speaking, Step 5 is not necessary: instead of computing the noisy count of $M(q, D)$, we could release the noisy count that was computed when deciding whether to release query q in Step 4. Re-using the noisy count from Step 4 would lead to all released query occurrence counts being skewed towards the larger side; for instance, there will be no queries whose reported number of occurrences is less than K. On the other hand, skipping Step 5 would improve the

[4]Assuming we desire to minimize the noise added and that $e^{1/b} \geq 1 + \frac{1}{2e^{(K-1)/b} - 1}$, which is the case for value ranges considered.

ϵ in the differential privacy guarantee of the algorithm by reducing it by d/b_c. It is up to the data releaser to choose the trade-off between more evenly distributed query counts and privacy.

Negative counts. The addition of Laplace random noise might yield lower or higher frequency counts than the true counts, in some cases yielding negative query occurrence or query-click frequency counts. The negative counts released where positive counts are expected are counter-intuitive but do not pose a privacy risk, rather a usability risk. If desired, one can perform a post-processing step replacing all negative counts with zeros without impacting privacy (since any post-processing that does not rely on the original data preserves privacy).

Query reformulations. Finally, we visit the issue of releasing query reformulations, which are a valuable part of the search log, but are not directly handled by our algorithm. In fact, query reformulations can be produced in the specific case when the reformulation is a click on a query suggestion. In such a case, a click on a reformulation is treated as a click on a URL, since a surfaced suggestion is as public as a surfaced URL. To ensure privacy, we could modify Step 2, Limit User Activity to count the first d queries that were typed and treat clicks on reformulations in the same way as URL clicks. These reformulations would not be as powerful and rich as actual user reformulations since they are limited to what a search engine is already capable of suggesting.

7. EXPERIMENTAL RESULTS

There is no doubt that attempting to preserve user privacy when performing a search log data release will take a toll on the utility of the data released, and that an algorithm that aims to satisfy rigorous privacy guarantees will not be able to release datasets that are as useful as the ones obtained through ad-hoc approaches such as merely replacing usernames with random identifiers. However, the decrease in utility is offset by the ability to protect user privacy and the ability to avoid PR disasters and retain user trust in the search engine. In this section, we describe several simple properties of the query and clicks data that can be released by applying our proposed algorithm to the search logs of a major search engine and characterize their dependence on the parameters of the algorithm and other choices that need to be made by the data releaser. The aim of the properties that we study is to serve as a proxy for the utility of the data released, as performing full-scale evaluations of complex algorithms on this data is beyond the scope of the paper.

Our experiments suggest that in the absence of other provably private methods for data release, and considering that our approach closely mimics the one that others are anecdotally considering utilizing, our proposed algorithm could serve as a first step towards the eventual goal of performing provably private and useful search log data releases.

Experimental Setup. We obtained the full query logs from a major search engine. The information necessary for our experiments is the session information per user (in order to restrict to d queries per user) and the issued queries together with their clicked urls. We performed a marginal amount of cleaning of the logs by removing special sets of characters (e.g. extra white spaces or runaway quotes) from the queries. In our comparisons below, we considered queries to match regardless of the case in words.

7.1 Published Query Set Characteristics

In our first experiment, we seek to gain insight into the effect of privacy guarantees desired from the algorithm on the following two characteristics of the published query set:

Percent of distinct queries released, i.e. the ratio of the number of distinct queries released to the actual number of distinct queries for all users in the original dataset (without limitations on how many queries the user can pose). Our goal here is to capture how representative is the published query set of the real set in terms of the released queries.

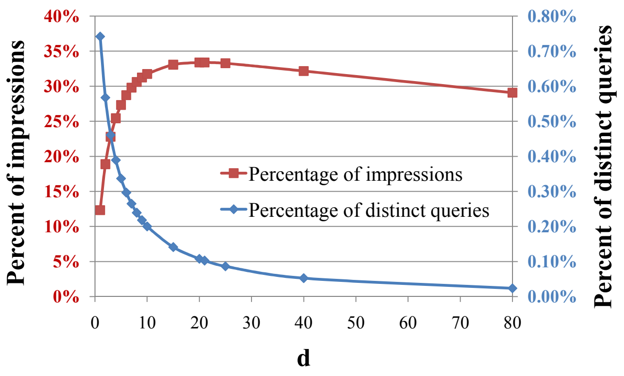

Figure 2: Percent of distinct queries and impressions released as a function of d, for fixed privacy parameters, $e^\epsilon = 10, \delta = 10^{-5}$ and one week time period

Percent of query impressions released, i.e. the ratio of the number of query impressions released (with each query accounted for as many times as the released noisy count) to the actual number of query impressions for all users in the original dataset. Our goal in this case is to capture how much query volume is published by our algorithm.

7.1.1 Effect of maximum queries d per user

A crucial step of our algorithm is to limit the number of queries posed per user that we consider for the release to d. Since the optimal choice of d is non-obvious from the perspective of data releaser, we start by studying the effect of d and fixed privacy parameters on the published query set characteristics when starting from a one week log from October 2007.

We compute the percent of distinct queries and impressions released by Algorithm 1 for different values of d and for parameters $e^\epsilon = 10$ and $\delta = 10^{-5}$, choosing the threshold K and noise b as described in Section 6. The results are shown in Figure 2. The horizontal axis represents the increasing values of d, the right vertical axis represents the percent of distinct queries released for a given value of d and the left vertical axis shows the percent of impressions for the respective d.

From Figure 2, we observe that although we can only publish a small percent of distinct queries overall, we can cover a reasonably large percent of impressions. More specifically, the output of our algorithm contains in total at most about 0.75% (for $d = 1$) of the distinct queries present in the original query log. However, the released queries correspond to about 10% – 35% of the search volume, depending on the choice of d, with a maximum of about 34% of volume achieved by d of around 20. The distinct queries released form a tiny fraction of the log because an overwhelming fraction of queries are issued very few times and our algorithm throws away such "tail queries" in order to guarantee privacy. However, we can release all the "frequent queries" and by virtue of their frequency, the volume of the released queries is substantial. Furthermore, although the percent of distinct queries released is small, the absolute number of the distinct queries released is large.

In Figure 2, we also observe that the percent of distinct queries released decreases as d increases. This is due to the fact that the threshold K increases more than linearly with d; when d increases, our dataset may contain more distinct queries and larger counts for

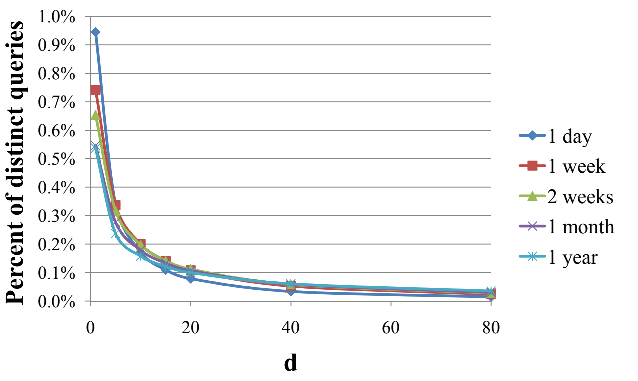

Figure 3: Percent of distinct queries released as a function of d for different time periods, with fixed privacy parameters, $e^\epsilon = 10, \delta = 10^{-5}$

all previously present queries, but such larger count is insufficient to offset the required increase in K to ensure the same privacy guarantees. Finally, we observe that for the one week log, the percent of impressions released initially increases with d, peaks for d around 20 and then decreases. There are two competing forces responsible for this observation: as d gets larger, more queries (and hence impressions) per user are included in the data while at the same time the threshold K needs to be increased in order to maintain the same privacy guarantees.

7.1.2 *Effect of time-span of the log*

We next study the effect of the time period of the log considered for release on the size of the released dataset. We repeated the previous experiment with query logs extracted over different time periods (one day, two weeks and one month from October 2007, and also one year) and compared to the output data generated from our log of one week. We plot the percent of released queries (distinct and impressions) in Figures 3 and 4 respectively. Figure 3 demonstrates that the percent of distinct queries released is more or less independent of the time-span of the source query log. On the other hand, the total number of query impressions does depend on the time-span as shown in Figure 4. In this case, we observe that the value of d at which the maximum percent of impressions is achieved increases with the length of the time period. This fits well with the intuition for sensible choices of d - as the time-span increases, the number of queries each user is limited to should increase as well.

The absolute number of queries (distinct and impressions) released increases with the increase in time-span of the source log. For example, for $d = 20$, the absolute number of distinct queries released grows 6-fold over one week, 12-fold over two weeks, 24-fold over one month, and 184-fold over one year time-spans of the source logs compared to that of a one day source log. Similarly the absolute number of impressions released grows 7-fold over one week, 15-fold over two weeks, 33-fold over one month, and 325-fold over one year durations compared to that of a one day source log. Thus, for the fixed choices of privacy parameters and d, it may be desirable to start with a query log extracted over a longer period of time, such as one year, to obtain better utility.

7.1.3 *Effect of privacy parameters*

We now turn to studying how the choice of multiplicative and additive privacy parameters used in Algorithm 1 affects the size of the released query set for fixed time period (one week) and $d = 21$[5].

[5]Although 21 queries per week per user seems to be a harsh restriction on the amount of queries posed, it turns out that an average user performs even fewer than 21 queries per week [12].

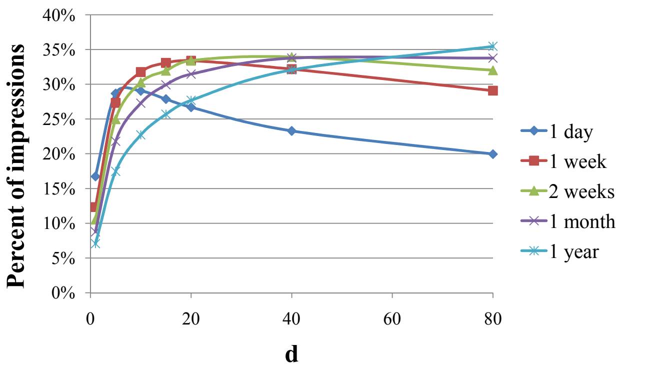

Figure 4: Percent of query impressions released as a function of d for different time periods, with fixed privacy parameters, $e^\epsilon = 10, \delta = 10^{-5}$

% of distinct queries	$\epsilon = \ln(2)$	$\epsilon = \ln(5)$	$\epsilon = \ln(10)$
$\delta = 10^{-6}$	0.03%	0.06%	0.09%
$\delta = 10^{-5}$	0.03%	0.07%	0.10%
$\delta = 10^{-4}$	0.04%	0.09%	0.12%

Table 2: Percent of distinct queries released as a function of privacy parameters, for one week time period and $d = 21$

Intuitively, the percent of distinct queries released should increase with less strict privacy requirements. More specifically, the larger the values of ϵ and δ, the larger portion of the original query log we should be able to release.

We plot the percent of distinct queries and impressions as a function of privacy requirements captured by different values of ϵ and δ, in Tables 2 and 3 respectively. They show that, in general, the percent of queries (distinct and impressions) that we can release increases as δ increases, with the relative increase in the percent of distinct queries being slightly higher for a given ϵ.

7.2 Utility of the Published Dataset

In this section our goal is to study the utility of the published dataset both in terms of the utility of the queries and the utility of the query click graph. First we give anecdotal examples of queries that could be released and then study the usefulness of published queries for social science research and the usefulness of the published query click graph for an algorithmic application.

We start with the query log over one year duration (restricted to $d = 21$ queries per user), and run **Release-Data** algorithm to obtain the queries safe to release, their noisy counts, and the noisy query-URL click counts. For each of the queries, we obtain the top 20 most clicked URLs instead of the top 20 URLs returned by the search engine, which is nearly equivalent as almost all users look at only the first page of 10 results [15], and very rarely beyond the second page and hence these URLs get the most clicks.

% of query impressions	$\epsilon = \ln(2)$	$\epsilon = \ln(5)$	$\epsilon = \ln(10)$
$\delta = 10^{-6}$	26.79%	30.92%	32.64%
$\delta = 10^{-5}$	27.55%	31.67%	33.38%
$\delta = 10^{-4}$	28.45%	32.55%	34.23%

Table 3: Percent of query impressions released as a function of privacy parameters, for one week time period and $d = 21$

Rank	Comorbidity Survey	Original Log	Released Queries
1.	Bugs, mice, snakes	Flying	Flying
2.	Heights	Heights	Heights
3.	Water	Snakes, spiders	Public speaking
4.	Public transportation	Death	Snakes, spiders
5.	Storms	Public speaking	Death
6.	Closed spaces	Commitment	Commitment
7.	Tunnels and bridges	Intimacy	Abandonment
8.	Crowds	Abandonment	The dark
9.	Speaking in public	The dark	Intimacy

Table 4: Most common fears, depending on the data source

For simplicity of experiment implementation, we did not limit the number of clicks per user, since most users click on very few results per search query anyway [12]. For determining the queries that are safe to release, we use the privacy parameters $e^{\epsilon} = 10$ and $\delta = 10^{-5}$ (so that the threshold $K = 147.4$ and noise $b = 9.1$). For click counts we use noise $b_c = 0.43$.

Some examples of non-trivial queries released are: "girl born with eight limbs", "cash register software", "vintage aluminum christmas trees", "how to tie a windsor knot".

7.2.1 *Studying Human Nature*

Since users communicate with a search engine in an uninhibited manner, posing queries containing their most private interests and concerns, the search log could also be an invaluable source of insight into human nature. We take an example proposed by Tancer [24] of using the queries posed by users to obtain insight into human fears and compare the conclusions that can be obtained by studying the queries of the original log vs the released queries.

Tancer [24] suspected that the insight one can gain into human fears through search logs is different than the data obtained through surveys. He compared the results of the National Comorbidity Survey [18], a phone survey where respondents were asked about their fears, with all Internet searches that contained the term "fear of", based on the theory that some people must be searching the Web to understand their fears. He observed that after removing those terms that are not phobia searches (such as "Fear of Clowns", a movie), the ordering of the most frequent fears as reported in the Comorbidity Survey differs from that obtained from searches; furthermore, there are more social fears in the list of top searches than in the list of top fears from the Comorbidity Survey.

We repeat Tancer's experiment on the search engine data available to us and on the perturbed release of that data, and obtain the ranking of fear frequencies that can be seen in Table 4.

The ordering of most popular fears is not fully preserved in the queries released compared to the original searches due to the noise added. However, the set of top nine fear searches obtained is the same in both cases, and is noticeably different from the one reported in the survey. For example, there is only one social fear in the Comorbidity Survey top list, the fear of speaking publicly, versus four in the "fear of" searches: public speaking, commitment, intimacy, and abandonment. Both the original log and the perturbed query click graph suggest that survey responses may not be truly reflecting what people are afraid of, and suggest a strong direction for further study by social scientists. Hence the published dataset could have immense utility in identifying directions for future study by social scientists and providing preliminary support for hypotheses.

7.2.2 *Keyword Generation*

We study the utility of the released query click graph by comparing the performance of an important application that utilizes a query click graph in its original and released forms. The application we study is *keyword generation*: given a business that is interested in launching an online advertising campaign around a concept, suggest keywords relevant to it. The idea of generating additional keywords from a seed set of keywords or URLs is powerful because it enables the advertisers to expand their exposure to users through bidding on a wider set of keywords. We use the algorithm proposed by Fuxman et al [13] that exploits the query click graph. Their algorithm takes a seed set of URLs about the concept and uses the idea of random walks on the query click graph with seed set URLs as absorbing states in order to generate more keywords. Typically, random walks are highly sensitive to changes in the graph, and hence, on one hand, it would be surprising if the keyword generation algorithm worked well on the released perturbed query click graph, given how much it has changed from the original query click graph. On the other hand, we would like to understand to what extent it still works.

We compare the keywords generated by this algorithm over the original graph and the released graph for three different seed sets. Each seed set consists of all URLs from the respective domains: 1) *shoes*: www.shoes.com; 2) *colleges*: Homepage domains of the top ten U.S. universities according to the US News Report ranking from early 2009; 3) *suicide*: six domains associated with depression and suicide (depression.com, suicide.com, and the 4 corresponding ones from WebMD and Wikipedia). The parameters of the keyword generation algorithm are set as in [13] ($\alpha = 0.001, \gamma = 0.0001$). We pruned all query-URL pairs with less than 10 clicks in both the original and the released graphs, for efficiency of implementation.

Compared to the query click graph based on the original log, the released private query click graph contains 9.85% of the queries and 20.43% of the edges of the original graph. Nonetheless, as can be seen in Table 5, we find that for all three seed sets, the absolute number of keywords generated by the algorithm using the released query click graph is substantial. For the shoes seed set, the number of keywords generated on the basis of the private graph comprises 13.9% to 24.9% of the number of keywords that can be generated using the original graph, depending on the relevance probability threshold used. For the college seed set the number of keywords generated is in the range of 12.9% to 16.1% of the original; and for the suicide seed set it is 4.4% to 7.4%. Moreover, more than 78% of the keyword suggestions obtained from the released graph were also obtained from the original graph, which is an indication of the similar level of relevance of the keywords produced in both cases. We observe greater overlap for more focused concepts (shoes: 93%, suicide: 99%).

We conclude that the released query click graph is still very useful for keyword generation. While one cannot generate as many keyword suggestions on the basis of the released graph as one would on the basis of the original graph[6], the volume of keyword suggestions generated is still substantial.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we took a first major step towards a solution for releasing search logs by proposing an algorithm for releasing queries and clicks in a manner that guarantees user privacy according to a rigorous privacy definition.

While we have shown that some non-trivial fraction of queries and impressions can be privately released and the released query

[6]It is important to note that the keyword suggestions obtained from the original graph that are not present among those obtained from the released graph, are not necessarily private. Algorithm 1 is conservative and chooses not to release many queries, a large fraction of which are likely not sensitive by themselves.

URL seed set		Relevance probability threshold 0.5	0.7	0.9	0.95
colleges	released	3,667	2,403	1,337	883
	original	28,346	17,103	8,287	6,373
shoes	released	2,733	1,620	448	248
	original	19,669	8,418	1,800	1,047
suicide	released	175	116	50	39
	original	3,945	2,517	806	525

Table 5: Number of keyword suggestions generated depending on URL seed set and query click graph source. Relevance probability refers to the probability that the keyword belongs to the seed set concept.

click graph can be successfully used for applications such as keyword generation and studies of people's fears, the question of whether this graph would be useful for other applications or whether it could serve as a benchmark log is far from being answered and is an interesting avenue for future work.

It is worth noting that as people invent more ways to group similar queries (such as "mom" and "mother"), we could use the grouping techniques to improve the performance of Algorithm 1.

A separate issue is that our algorithm implicitly assumes that users behave in an honest manner. However, there are ways for an attacker to maliciously bring private tail queries into the head. For instance, since K, b, d are public parameters, an attacker could create, say, $K + 5b$ copies of themselves and in their first d queries issue a private query such as someone else's credit card number. The net effect is that the search engine would publish this private data. We do not have a way to get around such malicious activities and leave this too as a direction for future work.

It seems promising to try using the query click graph to generate a synthesized search log: select a query according to the perturbed frequency distribution that we publish, select clicks according to the perturbed probability a document is clicked, select a reformulation according to the perturbed distribution over reformulations, or select a new query. This procedure would not generate a search log per se, since no timestamps would be published and it is not clear if the sessions would actually be meaningful. We leave open the question of how best to generate a search log from the perturbed data that we publish.

Finally, there are many other aspects to releasing search logs besides the privacy of users. For instance, releasing queries and clicks reveals at a large scale the performance of a search engine. Thus, the log leaks queries where the search engine performs poorly, e.g., abandoned head queries or head queries with few clicks. As another example, search data reveals queries that surface adult content when they should not. So beyond the privacy of users, there are other factors to consider before search data is released.

9. ACKNOWLEDGMENTS

We thank Rakesh Agrawal and Frank McSherry for valuable insights and discussions, Ariel Fuxman, Alan Halverson, and Panayiotis Tsaparas for help with experimental setup, and Nick Craswell for thoughtful feedback.

10. REFERENCES

[1] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Query Log Analysis: Social And Technological Challenges Workshop at WWW*, 2007.

[2] M. Arrington. AOL proudly releases massive amounts of private data. August 2006.

[3] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *KDD*, pages 76–85, 2007.

[4] J. Bar-Ilan. Access to query logs - an academic researcher's point of view. In *Query Log Analysis: Social And Technological Challenges Workshop at WWW*, 2007.

[5] M. Barbaro and T. Zeller. A face is exposed for AOL searcher No. 4417749. *New York Times*, Aug 2006.

[6] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[7] K. Chaudhuri and N. Mishra. When random sampling preserves privacy. In *CRYPTO*, volume 4117, pages 198–213, 2006.

[8] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, pages 239–246, 2007.

[9] C. Dwork. An ad omnia approach to defining and achieving private data analysis. In *Lecture Notes in Computer Science*, volume 4890, pages 1–13. Springer, 2008.

[10] C. Dwork, K. . Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, volume 4004, pages 486–503, 2006.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284, 2006.

[12] D. Fallows. Search engine users. *Pew Internet and American Life Project*, 2005.

[13] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *WWW*, pages 61–70, 2008.

[14] A. Horowitz, D. Jacobson, T. McNichol, and O. Thomas. 101 dumbest moments in business, the year's biggest boors, buffoons, and blunderers. In *CNN Money*, 2007.

[15] T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*, pages 154–161, 2005.

[16] R. Jones, R. Kumar, B. Pang, and A.Tomkins. "I know what you did last summer": query logs and user privacy. In *CIKM*, pages 909–914, 2007.

[17] R. Jones, R. Kumar, B. Pang, and A. Tomkins. Vanity fair: Privacy in querylog bundles. In *CIKM*, pages 853–862, 2008.

[18] R. Kessler, M. Stein, and P. Berglund. Social Phobia Subtypes in the National Comorbidity Survey. *Am J Psychiatry*, 155(5):613–619, 1998.

[19] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On anonymizing query logs via token-based hashing. In *WWW*, pages 629–638, 2007.

[20] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[21] F. McSherry and K. Talwar. Private communication. 2008.

[22] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[23] K. Nissim. Private data analysis via output perturbation. In *Privacy-Preserving Data Mining: Models and Algorithms*, pages 383–414. Springer, 2008.

[24] B. Tancer. *Click: What Millions of People Are Doing Online and Why it Matters*. Hyperion, 2008.

[25] L. Xiong and E. Agichtein. Towards privacy-preserving query log publishing. In *Query Log Analysis: Social And Technological Challenges Workshop in WWW*, 2007.

11. APPENDIX

Observation 2. **[Properties of Laplace distribution]** For Laplace distribution with location parameter 0, and scale parameter $b > 0$, and a random variable X, the cdf $F(x) = Pr[X \leq x]$ satisfies:

$$\begin{aligned} F(x) &= 1/2 \cdot \exp(x/b),\ if\ x < 0 \\ &= 1 - 1/2 \cdot \exp(-x/b),\ if\ x \geq 0 \end{aligned}$$

In this notation, increasing b flattens out the $Lap(b)$ curve, yielding larger expected noise magnitude and therefore, eventually, better privacy guarantees.

Observation 3. **[Properties of Laplace ratios]** Let r be a random $Lap(b)$ noise. Then,
$1 \leq \frac{Pr[r<c+1]}{Pr[r<c]} \leq e^{1/b}$ and $e^{-1/b} \leq \frac{Pr[r>c+1]}{Pr[r>c]} \leq 1$.

Observation 4. **[Properties of ratios]** For $a, b \geq 0$ and $c, d > 0 : \frac{a+b}{c+d} \leq \max(\frac{a}{c}, \frac{b}{d})$.