Privacy in Search Logs

Michaela Götz       Ashwin Machanavajjhala       Guozhang Wang       Xiaokui Xiao
Johannes Gehrke
April 4, 2009

Abstract

Search engine companies collect the “database of intentions”, the histories of their users’ search queries. These search logs are a gold mine for researchers. Search engine companies, however, are wary of publishing search logs in order not to disclose sensitive information.

In this paper, we develop a novel algorithm called ZEALOUS that for the first time enables publishing frequent keywords, queries, and clicks from a search log while achieving a very strong privacy guarantee called $(\epsilon, \delta)$-probabilistic differential privacy. An extensive experimental evaluation shows that search logs published with ZEALOUS can be used for research on both search quality and search efficiency with little loss in utility.

1 Introduction

Civilization is the progress toward a society of privacy. The savage’s whole existence is public, ruled by the laws of his tribe. Civilization is the process of setting man free from men. — Ayn Rand.

My favorite thing about the Internet is that you get to go into the private world of real creeps without having to smell them. — Penn Jillette.

It is hard to imagine the Internet today without the easy access to Internet Search Engines that help us to find pieces of information on the Web. Whenever a user submits a search query, the search engine logs the query and other information associated with it (for example, what links the user clicked on). The content of these search logs enable much valuable research within search engine companies to improve both search experience and search performance as exemplified in the following two applications:

- Search Performance: Index caching [2] is designed to reduce query response time by keeping parts of the inverted index in main memory while putting the rest in lower levels of the storage hierarchy (such as SSD or hard disks). The placement is computed based on access frequencies of keywords in the search log.

- Search Quality: Query Substitution [10] studies how to rephrase a user query to match it to documents or advertisements that do not contain the actual keywords of the query but contain relevant information. Algorithms for query substitution examine the subsequent query pairs in the search log to learn how users sequentially rephrase queries.

We expect that publishing search logs will boost the research on search quality and efficiency. Currently, this research is almost exclusively conducted at search engine companies. For example, in the last three years only a single paper was published at WWW that conducted search log analysis but did not have an author working for a search engine company. All other 15 papers on search log analysis can be linked to search engines through the list of authors. We expect this to change once all researchers have access to search logs. The reason why search engines are reluctant to publish their search logs is that they are concerned with user privacy disclosure.
In this paper, we investigate the problem of disclosure control when publishing search logs \[1, 12, 16\]. By disclosure control, we mean that although we publish (a modified) search log, an attacker will not be able to infer sensitive information about individuals whose search data is contained in the published log. We seek algorithms generating sanitized search logs that limit the disclosure of sensitive information. Prior events show that this is a hard problem. AOL published three months of search logs of 650,000 users where the only privacy protection was the replacement of user-ids with random numbers. However, the queries of a user usually contain identifying information such as searches for addresses and local events or even for the user’s name. This information can be linked to external databases to re-identify the user. The log that has been suitably termed the “database of intentions” then unravels the life and personality of the user through her searches for diseases, habits, lifestyle choices, personal tastes, and political affiliations. For example, the New York Times identified Ms. Thelma Arnold from Lilburn, Georgia as searcher number 4417749 \[3\] in the AOL search log; her queries contained not only enough information to identify herself, but also showed her love for dogs and her friends’ medical ailments. This data has now made the rounds on the Internet resulting in Internet websites that serve the data such as \texttt{aolpsycho.com}, as well as blog comments such as “Last week, AOL did another stupid thing, but at least it was in the name of science.” \[1\] More attacks have been proposed showing that the removal of names, age, zip codes and other identifiers does not prevent re-identification of the authors of queries \[9\]. Replacing keywords in search queries by random numbers is also not sufficient to guarantee privacy \[13\].

As a response, several algorithms based on \(k\)-anonymity have been proposed for sanitizing search logs \[1, 16\]. Conceptually, these algorithms guarantee that an entry in the published search log can only be linked to at least \(k\) other users. However, unfortunately these privacy definitions are too weak in practice. As we will discuss in Section \[6\] queries in a search log is different from generic set-valued attributes since adversaries can actively influence the search log by submitting queries themselves, and thus weak privacy definitions that are based only on anonymity leak too much private information.

**Our Contributions.** In this paper, we describe a novel way of publishing search logs that guards against very powerful adversaries, and we perform a thorough study of the utility of our published search logs in comparison with the original search logs. In particular, this paper makes the following contributions:

- **Strong Privacy Guarantees.** Our work enforces \((\epsilon, \delta)\)-probabilistic differential privacy \[15\]: We guarantee that an attacker learns roughly the same about a user whether or not the user’s search history was part of the published data. Providing very strong privacy guarantees is important for search logs, since attackers can manipulate the search log themselves by submitting search queries. Prior work fails to guard against such active attackers and thus provides insufficient privacy guarantees.

- **An Efficient Algorithm for Publishing Search Logs.** We develop a novel algorithm called ZEALOUS\[2\] that enables us to publish search logs while enforcing \((\epsilon, \delta)\)-probabilistic differential privacy. ZEALOUS can gracefully handle the huge domain and the sparsity of the data while preserving frequent keywords, queries and clicks.

- **A Thorough Study of Utility With Real Applications.** We evaluate the utility of ZEALOUS and compare it to related work. Instead of only comparing summary statistics between the original and published log, we take a novel, application-oriented approach to utility: We implement the two search log applications described in the first paragraph, run these applications on the original and the published sanitized search log, and compare the results using application-specific metrics. Our evaluation shows that the search log produced using ZEALOUS retains good utility despite its much stronger privacy guarantees compared to previous work.

\[1\] \url{http://en.wikipedia.org/wiki/AOL_search_data_scandal} has more information, including links to the resignation of AOL’s CTO and the ongoing class action lawsuit against AOL resulting from the data release.

\[2\] ZEArch LOg pUbliSher
We continue in Section 2 with some background on search logs and a discussion of the shortcomings of prior work on sanitizing search logs. Section 3 describes ZEALOUS and the analysis of its privacy guarantee. Section 4 gives an in-depth analysis of choosing parameters for ZEALOUS, and Section 5 experimentally evaluates ZEALOUS against prior work. We give an overview of previous work in Section 6, and conclude in Section 7.

2 Preliminaries

In this section, we will give a brief introduction on search logs and data privacy, providing the reader with the necessary terminology and background for Section 3 where we describe ZEALOUS.

2.1 Search Logs

Search engines such as Microsoft Live Search, Google, Yahoo, AskJeeves, or others enable users to ask keyword queries and return a ranked list of relevant websites. Users then click on one or more of these links. Search engines have sophisticated ways to identify users; for example, users may be logged on to accounts provided by the search engine, or they might be identified via cookies and IP address. A search log is a collection of search log entries that contain data about the users’ queries and the links that they clicked on. We assume that a search log entry has the following schema:

\[ \text{⟨user-id, query, time, clicks⟩} \]

where a user-id identifies a user, a query is a set of keywords, and clicks is a list of ⟨url, pos⟩ pairs, indicating that the user clicked on the given link (url) which was displayed at rank (pos) to the user. A user history consists of all search entries from a single user. Such a history is usually partitioned into sessions containing queries with similar user intent; many details go into this partitioning that are orthogonal to all the techniques in this paper. Query pairs are two subsequent queries from the same user that are contained in the same session.

Search engines compute histograms, or counts of keywords, queries, etc., from the search log, and use these instead of the actual log for a variety of applications. The algorithm we propose in the next section publishes noisy histograms rather than the whole search log. We define the keyword histogram of a search log \( SL \) as a set of pairs \((k, c_k)\). Here \( k \) is a keyword and \( c_k \) is the number of users in whose search history in \( SL \) contains that keyword. We define the query histogram, the query pair histogram, and the click histogram analogously.

2.2 Privacy

When thinking about privacy in search logs, we have to answer the following questions: (1) What information in the search log is sensitive? (2) What attackers do we consider? (3) What privacy guarantee should we give?

Sensitive Information. Due to the very sensitive content in search logs, we take a very conservative approach toward privacy: We consider all information in a search log to be sensitive. It is not only sensitive what query or keyword a user asked, it is also sensitive on what documents the user clicked.

Power of Attacker. We want to guarantee privacy against the largest possible class of attackers. An attacker can actively add data to the search log (simply by submitting queries to the search engine) and then she can use any external source of information to analyze the published search log. In particular, we want to protect privacy against active attacks, linking attacks, and the homogeneity attack without making any assumptions about the attacker’s background knowledge.

Privacy Guarantee. We want to guarantee that an attacker learns roughly the same information about a user whether or not the search history of that user was included in the published search log. With this guarantee a user does not regret having used the particular search engine for his or her queries. Dwork et al. turned this idea into a formal privacy definition called \( \epsilon \)-differential privacy that we apply here to search logs:
Definition 1 [8] An algorithm $A$ is $\epsilon$-differentially private if for all search logs $SL$ and $SL'$ differing in the search history of a single user and for all output search logs $S$:

$$Pr[A(SL) = S] \leq e^\epsilon Pr[A(SL') = S].$$

This definition ensures that a user has no reason to complain that the search engine published $S$, since $S$ could have also arisen from a search log $SL'$ that did not include the correct search history of the user. We will refer to search logs that only differ in the search history of a single user as "neighboring search logs" in the remainder of this paper.

A probabilistic version of differential privacy called $(\epsilon, \delta)$-probabilistic differential privacy that relaxes $\epsilon$-differential privacy has been proposed by Machanavajjhala et al.:

Definition 2 [10] An Algorithm $A$ satisfies $(\epsilon, \delta)$-probabilistic differential privacy if for all search logs $SL$ we can divide the output space into to sets $\mathcal{S}, \bar{\mathcal{S}}$ such that

1. $Pr[A(SL) \in \bar{\mathcal{S}}] \leq \delta$, and
2. for all neighboring search logs $SL'$ and for all $S \in \mathcal{S}$:
   $$Pr[A(SL) = S] \leq e^\epsilon Pr[A(SL') = S] \text{ and } Pr[A(SL') = S] \leq e^\epsilon Pr[A(SL) = S].$$

Definition 2 allows $A$ to be imperfect: With probability at most $\delta$ the output of $A$ could breach the privacy of an individual. But if we choose $\delta$ to be small then with high probability the output of $A$ offers the same privacy guarantee as $\epsilon$-differential privacy.

Note that we also considered another relaxation of $\epsilon$-differential privacy called $(\epsilon, \delta)$-indistinguishability $\checkmark$, which, intuitively, ensures that each user’s privacy is secured with a probability of at least $1 - \delta$. Nevertheless, $(\epsilon, \delta)$-indistinguishability does not provide any guarantee on the probability that the privacy of all users is protected. As a consequence, the chance that a privacy breach occurs for at least one user can be significantly larger than $\delta$. In contrast, $(\epsilon, \delta)$-probabilistic differential privacy is provably strictly stronger: it ensures that, with at least $1 - \delta$ probability, no privacy breach will occur for any user. History has shown, as in the case of Ms. Arnold, that breaching the privacy of a single user can be disastrous. Hence, we believe that the stronger notion of privacy is needed in practice.

3 The ZEALOUS Algorithm

This section presents ZEALOUS, a novel privacy-preserving algorithm for publishing histograms from a search log. ZEALOUS is efficient despite the sparsity of the data and the huge (infinite) domain of these items. ZEALOUS can be used to publish histograms of keywords, queries, query pairs or clicks. For ease of presentation, we explain in the remainder of this section the instance of ZEALOUS that publishes keywords; other instantiations are straightforward, and we will evaluate these other instances of ZEALOUS experimentally in Sections 4 and 5.

One of the advantages of ZEALOUS is its simplicity: It uses a two-step process to eliminate the tail of the search log, i.e., the keywords with low counts, to achieve a strong privacy guarantee. We give the pseudocode of ZEALOUS next; Figure 1 gives a pictorial description of ZEALOUS.

Algorithm ZEALOUS for Publishing a Keyword Histogram of a Search Log

Input: Search log $SL$, positive numbers $m$, $\lambda$, $\tau$, $\tau'$

1. For each user $u$ select a set $s_u$ of up to $m$ distinct keywords from $u$’s search history in $SL$.
2. Create the keyword histogram of pairs $(k, c_k)$ from the selected keywords, i.e. the histogram $= \{(k, c_k) | c_k = \sum_{u:k \in s_u} 1\}$.
3. Delete from the histogram the pairs $(k, c_k)$ with count $c_k < \tau$.
4. Add noise to each non-zero count. That is, for each pair $(k, c_k)$ in the histogram:
   Sample a random number $q_k$ from the distribution $\text{Lap}(\lambda)$ and add it to the count, resulting in a noisy count: $\tilde{c}_k \leftarrow c_k + q_k$.

$^3$The Laplace distribution with scale parameter $\lambda$ has the probability density function $\frac{1}{2\lambda}e^{-\frac{|x|}{\lambda}}$. 

4
5. Delete from the histogram the pairs \((k, \hat{c}_k)\) with noisy counts \(\hat{c}_k\) that are no more than \(\tau'\).

6. Publish the remaining keywords and their noisy counts.

The following theorem tells us how to set the parameters \(\lambda, \tau, \tau',\) and \(m\) such that ZEALOUS guarantees \((\epsilon, \delta)\)-probabilistic differential privacy.

**Theorem 3** Given a search log \(SL\) and positive numbers \(m, \tau, \tau',\) and \(\lambda\), ZEALOUS achieves \((\epsilon, \delta)\)-probabilistic differential privacy, if

\[
\lambda \geq 2m/\epsilon, \text{ and } \quad (1)
\]

\[
\tau' - \tau \geq \max\left(-\lambda \ln \left(2 - 2e^{-\frac{1}{\lambda}}\right), -\lambda \ln \left(\frac{2\delta}{U \cdot m/\tau}\right)\right), \quad (2)
\]

where \(U\) denotes the number of users in \(SL\).

**Proof 1** Given \(SL\), let \(H\) be the keyword histogram constructed by ZEALOUS in Step 2, and \(K\) be the set of keywords in \(H\) whose count equals \(\tau\). Let \(\Omega\) be the set of keyword histograms, that do not contain any keyword in \(K\). For notational simplicity, let us denote ZEALOUS as a function \(Z\). We will prove the theorem by showing that, given Equations 1, 2, and 3,

\[
Pr[Z(SL) \notin \Omega] \leq \delta, \quad (3)
\]

and for any keyword histogram \(\omega \in \Omega\) and any neighboring search log \(SL'\) of \(SL\),

\[
e^{-\epsilon} \cdot Pr[Z(SL') = \omega] \leq Pr[Z(SL) = \omega] \leq e^{\epsilon} \cdot Pr[Z(SL') = \omega]. \quad (4)
\]

We will first prove that Equation 3 holds. Assume that the \(i\)-th \((i \in [1, |K|])\) keyword in \(K\) has a count \(c_i\) in \(Z(SL)\). Then,

\[
Pr[Z(SL) \notin \Omega] \\
= Pr[\exists i \in [1, |K|], c_i > \tau'] \\
= 1 - Pr[\forall i \in [1, |K|], c_i \leq \tau'] \\
= 1 - \prod_{i \in [1, |K|]} \left(\int_{-\infty}^{\tau' - \tau} \frac{1}{2\lambda} e^{-\frac{x}{\lambda}} dx\right) \quad \text{(the noise added to } c_i \text{ has to be } \geq \tau' - \tau) \\
\leq \frac{|K|}{2} \cdot e^{-\frac{\tau' - \tau}{\lambda}} \quad \text{(because } |K| \leq U \cdot m/\tau) \\
\leq \frac{U \cdot m}{2\tau} \cdot e^{-\frac{\tau' - \tau}{\lambda}} \quad \text{(by Equation 2)} \\
= \delta. \quad (5)
\]

Next, we will show that Equation 4 also holds. Let \(SL'\) be any neighboring search log of \(SL\). Let \(\omega\) be any possible output of ZEALOUS given \(SL\), such that \(\omega \in \Omega\). To establish Equation 4, it suffices to prove that

\[
\frac{Pr[Z(SL) = \omega]}{Pr[Z(SL') = \omega]} \leq e^\epsilon, \quad \text{and} \quad (6)
\]

\[
\frac{Pr[Z(SL') = \omega]}{Pr[Z(SL) = \omega]} \leq e^\epsilon. \quad (7)
\]

As the proofs for Equations 6 and 7 are similar, in the following we will focus on Equation 6 for simplicity.
Given \( SL' \), let \( H' \) be the keyword histogram constructed by ZEALOUS in Step 2. Let \( \Delta \) be the set of keywords that have different counts in \( H \) and \( H' \). Since \( SL \) and \( SL' \) differ in the search history of a single user, and each user contributes at most \( m \) keywords, we have \( |\Delta| \leq 2m \). Let \( k_i \) (\( i \in [1, |\Delta|] \)) be the \( i \)-th keyword in \( \Delta \), and \( d_i, d_i' \), and \( d_i^* \) be the counts of \( k_i \) in \( H \), \( H' \), and \( \omega \), respectively. Since a user adds at most one to the count of a keyword (see Step 2.), we have \( d_i - d_i' = 1 \) for any \( i \in [1, |\Delta|] \). To simplify notation, let \( E_i, E_i' \), and \( E_i^* \) denote the event that \( k_i \) has counts \( d_i, d_i' \), and \( d_i^* \) in \( H, H' \), and \( Z(\cdot) \), respectively. It can be verified that

\[
\frac{Pr[Z(SL) = \omega]}{Pr[Z(SL') = \omega]} = \prod_{i \in [1, |\Delta|]} \frac{Pr[E_i^* \mid E_i]}{Pr[E_i^* \mid E_i']}
\]

In what follows, we will show that \( \frac{Pr[E_i^* \mid E_i]}{Pr[E_i^* \mid E_i']} \leq e^{1/\lambda} \) for any \( i \in [1, |\Delta|] \). We differentiate three cases: (i) \( d_i \geq \tau, d_i' \geq \tau \), (ii) \( d_i < \tau \) and (iii) \( d_i = \tau \) and \( d_i' = \tau - 1 \).

Consider case (i) when \( d_i \) and \( d_i^* \) are at least \( \tau \). Then, if \( d_i^* > 0 \), we have

\[
\frac{Pr[E_i^* \mid E_i]}{Pr[E_i^* \mid E_i']} = \frac{\frac{1}{\lambda} e^{-|d_i^* - d_i|/\lambda}}{\frac{1}{\lambda} e^{-|d_i' - d_i|/\lambda}} = e^{(d_i^* - d_i' - (d_i^* - d_i))/\lambda} \leq e^{d_i - d_i'}/\lambda = e^{\frac{1}{\lambda}} \quad \text{(because } |d_i - d_i'| = 1 \text{ for any } i \text{)}
\]

On the other hand, if \( d_i^* = 0 \),

\[
Pr[E_i^* \mid E_i] = \int_{-\infty}^{\tau - d_i} \frac{1}{2\pi} e^{-|x|/\lambda}dx = \int_{-\infty}^{\tau - d_i} \frac{1}{\sqrt{2\pi}} e^{-|x|/\lambda}dx \leq e^{\frac{1}{\lambda}}.
\]

Now consider case (ii) when \( d_i \) is less than \( \tau \). Since \( \omega \in \Omega \), and ZEALOUS eliminates all counts in \( H \) that are smaller than \( \tau \), we have \( d_i^* = 0 \), and \( Pr[E_i^* \mid E_i] = 1 \). On the other hand,

\[
Pr[E_i^* \mid E_i'] = \begin{cases} 1, & \text{if } d_i' \leq \tau \\ 1 - \frac{1}{2} e^{-|\tau - d_i'|/\lambda}, & \text{otherwise} \end{cases}
\]

Therefore,

\[
\frac{Pr[E_i^* \mid E_i]}{Pr[E_i^* \mid E_i']} \begin{cases} 1, & \text{if } d_i^* = 0 \\ \leq 1 - \frac{1}{2} e^{-|\tau - d_i'|/\lambda}, & \text{ otherwise} \end{cases}
\]

Lastly, consider case (iii) when \( d_i = \tau \) and \( d_i^* = \tau - 1 \). Since \( \omega \in \Omega \) we have \( d_i^* = 0 \). Moreover, since ZEALOUS eliminates all counts in \( H \) that are smaller than \( \tau \), it follows that \( Pr[E_i^* \mid E_i'] = 1 \). Therefore,

\[
Pr[E_i^* \mid E_i] = Pr[E_i^* \mid E_i'] \leq e^{\frac{1}{\lambda}}.
\]

In summary, \( \frac{Pr[E_i^* \mid E_i]}{Pr[E_i^* \mid E_i']} \leq e^{1/\lambda} \). Since \( |\Delta| \leq 2m \), we have

\[
Pr[Z(SL) = \omega] = \prod_{i \in [1, |\Delta|]} \frac{Pr[E_i^* \mid E_i]}{Pr[E_i^* \mid E_i']} = \prod_{i \in [1, |\Delta|]} e^{1/\lambda} = e^{|\Delta|/\lambda} \leq e < \lambda.
\]

Therefore, the theorem is proved.

### 3.1 Map–Reduce Implementation

Given a search log grouped by user-ids, ZEALOUS runs in time linear in the size of the search log, and it has a modest space complexity. However, we expect that ZEALOUS will be applied to search logs with billions of search requests, which may be distributed across hundreds of machines. To handle such a large
search log, we easily parallelize ZEALOUS through a map–reduce implementation. The parallel algorithm has two steps. In the first step we select at most \( m \) contributions per user through a Map–Reduce phase. In the second step, we generate the histogram, filter out items with counts below \( \tau \), add noise to the remaining counts, and output only items and their noisy count if it is above \( \tau' \); this again can be easily done through a Map–Reduce phase. The pseudocode of the algorithm is as follows.

\[
\text{Input:} (\text{USER-ID}, (\text{QUERY, TIME, CLICKS}))
\]

\[
\text{Map1} (\text{user-id, (query, time, clicks)}) \implies \text{list}((\text{user-id, keyw}_1), (\text{user-id, keyw}_2), \ldots)
\]

\[
\text{Reduce1} (\text{user-id, list(keyw}_1, \text{keyw}_2, \ldots)) \implies (\text{user-id, list(keyw}_1, \text{keyw}_2, \ldots))
\]

\[
\text{Map2} (\text{user-id, list(keyw}_1, \text{keyw}_2, \ldots, \text{keyw}_{\leq m})) \implies \text{list}((\text{keyw}_1, 1), (\text{keyw}_2, 1), \ldots, (\text{keyw}_{\leq m}, 1))
\]

\[
\text{Reduce2} (\text{keyw}_i, \text{list}(1, \ldots)) \begin{cases} (\text{keyw}_i, \hat{c}), & \text{if total count } \geq \tau, \\ \hat{c} = \text{total count} + \text{Lap}(\lambda), & \text{total count } \geq \tau' \\ \bot, & \text{o.w.} \end{cases}
\]

3.2 A Discussion of Alternatives

Now that we have seen ZEALOUS and its properties, we explain why we cannot simply apply prior work, and we also explain the rationale behind the various steps of ZEALOUS that address some of the true difficulties in publishing search logs while limiting disclosure of sensitive information.

Since we want to enforce differential privacy, we could first think about applying an existing algorithm to the search log. One existing algorithm for publishing histograms, developed by Dwork et al. [8], adds noise sampled from a Laplacian distribution to every keyword that a user could have used to formulate a query. The amount of noise that we have to add depends on the sensitivity \( s \) of the histogram, where the sensitivity is the maximum L1 difference between the histograms of two neighboring search logs. However, if we do not bound the size of the history of any user, then a single user could potentially affect the count of every keyword (consider for example a user whose search history contains every word in the English language or even random strings), thus leading to a very large value of \( s \). The corresponding amount of noise (\( \frac{2s}{\lambda} \)) would be so large that the output of this algorithm would retain no useful information in the output search log. Hence, we need to limit the number of keywords contributed by each user. Note that ZEALOUS implements exactly this restriction as Step 1.

After limiting the contributions per user we would still have to add noise to the count of every possible keyword that could have been used to formulate a query. For instance, even if we limit the size of keywords to at most 30 characters, the size of the domain of all possible keywords would be \( N = 26^{30} \!). We would need to add noise to each of these \( N \) keywords even if they do not exist in the search log and thus have a count of zero. It is clearly inefficient to add noise to such a huge number of counts (most of which are zeros anyway).

A tractable algorithm would only add noise to the keywords that actually occur in the search log, (or more generally to all counts above a threshold \( \tau \)). This version of the algorithm corresponds to steps 1 - 4 of ZEALOUS. However, as stated in the following proposition, adding noise to only the keywords with count above some threshold guarantees neither \( \epsilon \)-differential privacy nor \((\epsilon, \delta)\)-probabilistic differential privacy.

**Proposition 1** Given a \( \tau > 0 \), adding noise only to keywords with counts above \( \tau \) guarantees neither \( \epsilon \)-differential privacy nor \((\epsilon, \delta)\)-probabilistic differential privacy.

The proof can be found in Appendix A. In contrast to the above approach, ZEALOUS manages to achieve privacy by applying a second threshold \( \tau' \) in Step 5. In summary, previous work on publishing histograms in a way that preserves \( \epsilon \)-differential privacy cannot be efficiently applied to search logs because of the size of the domain. ZEALOUS handles the size of the domain gracefully: It focuses on publishing only
frequent items and thus works independently of the size of the domain.

4 Choosing Parameters

Apart from the privacy parameters $\epsilon$ and $\delta$, ZEALOUS requires the data publisher to specify two more parameters: $\tau$, the first threshold used to eliminate keywords with low counts (Step 3), and $m$, the number of contributions per user. These parameters affect both the noise added to each count as well as the second threshold $\tau'$. Before we discuss the choice of these parameters we explain the general set-up of our experiments.

**Data.** In our experiments we work with a search log of user queries from the Yahoo! search engine collected from 500,000 users over a period of one month. This search log contains about one million distinct keywords, three million distinct queries, three million distinct query pairs, and 4.5 million distinct clicks.

**Privacy Parameters.** In all experiments we set $\delta = 0.001$. Thus the probability that the output of ZEALOUS could breach the privacy of a user is appropriately small. We explore different levels of differential privacy by varying $\epsilon$.

4.1 Choosing Threshold $\tau$

We would like to retain as much information as possible in the published search log. A smaller value for $\tau'$ immediately leads to a histogram with higher utility because fewer items and their noisy counts are filtered out in the last step of ZEALOUS. Thus if we choose $\tau$ in a way that minimizes $\tau'$ we maximize the utility of the resulting histogram. Interestingly, choosing $\tau = 1$ does not necessarily minimize the value of $\tau'$.

Table 2 presents the value of $\tau'$ for different values of $\tau$ for $m = 2$ and $\epsilon = 1$.

As we can see, for our parameter setting $\tau'$ is minimized if $\tau = 4$. We can show the following strong optimality result which tells us how to choose $\tau$ optimally in order to maximize utility.

**Proposition 2** For fixed $\epsilon, \delta$ and $m$ choosing $\tau = \lceil 2m/\epsilon \rceil$ minimizes the value of $\tau'$.

The proof follows from taking the derivative of $\tau'$ as a function of $\tau$ (based on Equation (2)) to determine its minimum.

4.2 Choosing the Number of Contributions $m$

Proposition 2 tells us how to set $\tau$ in order to maximize utility. However, it is less clear how to set $m$ optimally. To discuss our choice of $m$ we will show the effect of varying $m$ on the coverage and the precision of the sanitized histogram of items. The top-$j$ coverage of a sanitized search log is defined as the fraction of distinct items among the top-$j$ most frequent items in the original search log that also appear in the sanitized search log. The top-$j$ precision of a sanitized search log is defined as the distance between the relative frequencies in the original search log versus the sanitized search log for the top-$j$ most frequent items. In particular, we study two distance metrics between the relative frequencies: the average L-1 distance and the KL-divergence.

As a first study of the coverage Table 2 shows the number of distinct items (recall that items can be keywords, queries, query pairs, or clicks) in the sanitized search log as $m$ increases. We observe that coverage decreases as we increase $m$. Moreover, the decrease in the number of published items is more dramatic for larger domains than for smaller domains. The number of distinct keywords decreases by 55% while at the same time the number of distinct query pairs decreases by 96% as we increase $m$ from 1 to 40.

This trend has two reasons. First, from Theorem 3 and Proposition 2 we see that threshold $\tau'$ increases
super-linearly in $m$. Second, as $m$ increases the number of keywords contributed by the users increases only sub-linearly in $m$; fewer users are able to supply $m$ items for increasing values of $m$. Hence, fewer items pass the threshold $\tau'$ as $m$ increases. The reduction is larger for query pairs than for keywords, because the average number of query pairs per user is smaller than the average number of keywords per user in the original search log (shown in Table 3).

| $\tau$ | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $\tau'$ | 81.1205 | 79.3479 | 78.7260 | 78.5753 | 78.6827 | 78.9534 | 79.3368 | 79.8027 | 80.3316 | 80.9101 |

Table 1: $\tau'$ as a function of $\tau$ for $m = 2$, $\epsilon = 1$, $\delta = 0.01$

The reduction is larger for query pairs than for keywords, because the average number of query pairs per user is smaller than the average number of keywords per user in the original search log (shown in Table 3). This suggests that one should never choose $m$ to be larger than the average number of items, because it decreases both the coverage and the total count while at the same time increases the noise.

Let us take a closer look at precision and coverage of the histograms of the various domains in Figures 2 and 3. In Figure 2 we vary $m$ between 1 and 40. Each curve plots the precision or coverage of the sanitized search log at various values of the top-$j$ parameter in comparison to the original search log. Based on the total number of distinct items in the original search log for the various domains we chose the values for the top-$j$ parameter appropriately. The upper two rows plot precision curves for the average L-1 distance (first row) and the KL-divergence (second row) of the relative frequencies. The lower two rows plot the coverage curves, i.e. the total (relative, respectively) number of top-$j$ items in the original search log that do not appear in sanitized search log in the third row (fourth row, respectively).

First, observe that the coverage decreases as $m$ increases, which confirms our discussion about the number of distinct items. Moreover, we see that the coverage gets worse for increasing values of the top-$j$ parameter. This illustrates that ZEALOUS gives better utility for the more frequent items. Second, note that for small values of the top-$j$ parameter, values of $m > 1$ give better precision. However, when the top-$j$ parameter is increased, $m = 1$ gives better precision because the precision of the top-$j$ values degrades due to items no longer appearing in the sanitized search log due to the increased cutoffs.

Figure 3 shows the same statistics varying the top-$j$ parameter on the x-axis. Each curve plots the precision for $m = 1, 2, 4, 8, 10, 40$, respectively. Note that $m = 1$ does not always give the best precision; for keywords, $m = 8$ has the lowest KL-divergence, and for queries, $m = 2$ has the lowest KL-divergence.

As we can see from our evaluation of these results,
there are two “regimes” for setting the value of $m$. If we are mainly interested in coverage, then $m$ should be set to 1. However, if we are only interested in a few top-$j$ items then we can increase precision by choosing a larger value for $m$, and we recommend the average number of items per user.

We will see this dichotomy again in our real applications of search log analysis: The index caching application does not require high coverage because of its storage restriction. However, the precision of the top-$j$ most frequent items is necessary to determine which of them to keep in memory. On the other hand, query substitutions are better with a larger number of distinct queries and query pairs because the number of substitution it generates depends on this coverage. Thus $m$ should be set to a large value for index caching and to a small value for query substitution.
5 Application-Oriented Evaluation

In this section, we experimentally evaluate ZEALOUS against a representative $k$-anonymity algorithm for publishing search logs. Traditionally, the utility of a privacy-preserving algorithm has been evaluated by comparing the input of the algorithm with the output to see "how much information is lost" by comparing some statistics of the sanitized output with the original data. The choice of suitable statistics is a difficult problem as these statistics need to mirror the sufficient statistics of applications that will use the sanitized search log. As notable differences, Brickell et al. [4] measure the utility with respect to data mining tasks and Kifer and Gehrke [11] develop specific techniques to boost utility with respect to log-linear models. In order to fully understand the performance of ZEALOUS in an application context, we picked two real applications from the information retrieval community to evaluate the utility of ZEALOUS: Index caching as a representative application for search performance, and query substitution as a representative application for search quality.

$k$-Query Anonymity. We will compare ZEALOUS against a representative $k$-anonymity algorithm for search logs called $k$-query anonymity [1]. We create a $k$-query anonymous search log as follows: We filter out all queries that are posed by fewer than $k$ distinct users. We then compute histograms of keywords, queries, and query pairs from the $k$-query-anonymous search log. We would like to emphasize that despite repeated requests over the several months, we were unable to obtain the code for $k$-session anonymity from the authors [16]. We will include a comparison with it in the final

Figure 3: Effect on Statistics Of Varying $j$ in top-$j$ for Different Values of $m$. 

(a) Keywords (b) Queries (c) Clicks (d) Query Pairs
that it is the goal of our comparison just to get a ball-
park understanding how much utility we lose in com-
parison to an approach based on $k$-anonymity. We will show using Examples 1 and 6 in Section 6 that the privacy guarantees provided by $k$-anonymity-
ated approaches are much weaker compared to our approach. In our comparison we seek to answer the question: What is the price in utility that we have to pay for a strong privacy guarantee?

We first describe our utility evaluation with statistics in Section 5.1 and then with real applications in Sections 5.2 and 5.3.

5.1 General Statistics
We explore different statistics that measure the difference of sanitized histograms to the histograms computed using the original search log. We analyze the histograms of keywords, queries, and query pairs for both sanitization methods. For clicks we only consider ZEALOUS histograms since a $k$-query anonymous search log does not contain any click data.

In our first experiment we compare the distribution of the counts in the histograms. Note that a $k$-query anonymous search log will never have query and key-
word counts below $k$, and similarly a ZEALOUS his-
togram will never have counts below $\tau'$. We choose $\epsilon = 5$, $m = 1$ for which threshold $\tau' \approx 10$. Therefore we deliberately set $k = 10$ such that $k \approx \tau'$ for a comparable setting.

Figure 4 shows the distribution of the counts in the histograms on a log-log scale. We see that the powerlaw shape of the distribution is well preserved. However, the total frequencies are lower for the sanitized search logs than the frequencies in the original histogram because our sanitization methods filter out user contributions. We also see the cutoffs created by $k$ and $\tau'$. One interesting observation is that as the domain increases from keywords to clicks and query pairs, the number of infrequent items becomes larger for the original search log. For example, the number of clicks with count one is an order of magnitude larger than the number of keywords with count one despite the fact that there are more keywords in the search log than clicks.

While it is good to know that the shape of the count distribution is well preserved, we would also like to know whether the counts of frequent keywords, queries, query pairs, and clicks are also preserved and what impact the privacy parameters $\epsilon$ and the anonymity parameter $k$ have.

Figure 5 shows the average differences to the counts in the original histogram. More precisely, we scaled up the sanitized histograms such that the total counts were equal to the total counts of the original his-
togram, then we calculated the average difference be-
tween the counts. The average is taken over all key-
words that have non-zero count in the original search log. As such this metric takes both coverage and pre-
cision into account.

As expected, with increasing $\epsilon$ the average difference decreases, since the noise added to to each count decreases. Similarly, by decreasing $k$ the accuracy increases, because more queries will pass the thresh-
old. Figure 5 shows that the average difference is comparable for the $k$ anonymous histogram and our ZEALOUS histogram. For keywords we observe that the ZEALOUS histogram is more accurate than a $k$ anonymous histogram for all values of $\epsilon > 2$. For queries we obtain roughly the same average difference for $k = 60$ and $\epsilon = 6$. For query pairs the $k$-query anonymous histogram provides better utility.

We also computed other metrics such as the root-
mean-square value of the differences and the total variation difference; they all reveal similar qualita-
tive trends. Thus despite the fact that ZEALOUS disregards many search log records (by throwing out all but $m$ contributions per user and by throwing out low frequent counts), ZEALOUS is able to preserve the overall distribution well.

5.2 Index Caching
Search engines maintain an inverted index which, in its simplest instantiation, contains for each keyword a posting list of identifiers of the documents in which the keyword appears. This index can be used to an-
swer search queries, but also to classify queries for choosing sponsored search results. While the index is often too large to fit in memory, maintaining a
Figure 4: Distributions of counts in the histograms.

Figure 5: Average difference between counts in the original histogram and the probabilistic differential privacy-preserving histogram, and the anonymous histogram for varying privacy / anonymity parameters $\epsilon$ and $k$. Parameter $m$ is fixed to 1.

part of it in memory reduces response time for all these applications. In the index caching problem, we aim to store in memory a set $S$ of posting lists that maximizes the hit-probability over all keywords (the formulation of the problem is from Baeza–Yates [2]). Given such a set $S$ and a probability distribution over the likelihood of occurrence of keywords in a query, the hit-probability is the sum of the likelihoods of the keywords whose posting list are kept in memory.

In our experiments, we use an improved version of the algorithm developed by Baeza–Yates [2] to decide which posting lists should be kept in memory. This algorithm first assigns each keyword a score, which equals its frequency in the search log divided by the number of documents that contain the keyword. Keywords are chosen using a greedy bin-packing strategy where we sequentially add posting lists from the keywords with the highest score until the memory is filled. In our experiments we fixed the memory size to be 1GB, and each document posting to be 8 Bytes (other parameters give comparable results). Our inverted index stores the document posting list for each keyword sorted according to their relevance which allows to retrieve the documents in the order of their relevance. We truncate this list in memory to contain at most 200,000 documents. Hence, for an incoming query the search engine retrieves the posting list for each keyword in the query either from memory or from disk. If the intersection of the posting lists happens to be empty, then less relevant documents are retrieved from disk for those keywords for which only the truncated posting list is kept on memory.

Figure 6 shows the hit–probabilities of the inverted index constructed using the original search log, the $k$-anonymous search log, and the ZEALOUS histogram with our greedy approximation algorithm. Figure 6(a) shows that our ZEALOUS histogram achieves better utility than the $k$-anonymous search
Whereas the hit probability for $m = 6$ is above 0.36 whereas the hit probability for $m = 1$ is less than 0.33. This confirms our findings about setting the value $m$ from Figure 2: For this application, the sanitized search log that are also contained in our ground truth ranking.

As a last experiment we study the effect of varying $m$ on the hit-probability in Figure 6(b). We observe that the hit probability for $m = 6$ is above 0.36 whereas the hit probability for $m = 1$ is less than 0.33. This confirms our findings about setting the value $m$ from Figure 2. For this application, the sanitized data should accurately model the relative frequencies of the most frequent keywords in the original search log, and thus a larger value of $m$ gives more accurate estimates of the hit-probability.

**5.3 Query Substitution**

Query Substitution studies how to rephrase a user query to match it to documents or advertisements that do not contain the actual keywords of the query but contain relevant information. Query substitution has applications in query refinement, sponsored search, and spelling error correction, just to name a few. Algorithms for query substitution examine query pairs to learn how users re-phrase queries. In the algorithm by Jones et al. [10], related queries for a query are identified in two steps. First, the query is partitioned into subsets of keywords, called *phrases*, based on their mutual information. Next, for each phrase, candidate query substitutions are determined based on the distribution of queries.

We run this algorithm to generate ranked substitution on the sanitized search logs. We then compare these rankings with the rankings produced by the original search log which serve as ground truth.

To measure the quality of the query substitutions, we compute the precision/recall, MAP (mean average precision) and NDG (normalized discounted cumulative gain) of the top-$j$ suggestions for each query; let us define these metrics next.

Consider a query $q$ and its list of top-$j$ ranked substitutions $q'_0, \ldots, q'_{j-1}$ computed based on a sanitized search log. We compare this ranking against the top-$j$ ranked substitutions $q_0, \ldots, q_{j-1}$ computed based on the original search log as follows. The *precision* is the fraction of substitutions from the sanitized search log that are also contained in our ground truth ranking:

$$
\text{Precision}(q) = \frac{|\{q_0, \ldots, q_{j-1}\} \cap \{q'_0, \ldots, q'_{j-1}\}|}{|\{q'_0, \ldots, q'_{j-1}\}|}
$$

Note, that the number of items in the ranking for a query $q$ can be less than $j$. The *recall* is the fraction of substitutions in our ground truth that are contained in the substitutions from the sanitized search log:

$$
\text{Recall}(q) = \frac{|\{q_0, \ldots, q_{j-1}\} \cap \{q'_0, \ldots, q'_{j-1}\}|}{|\{q_0, \ldots, q_{j-1}\}|}
$$

MAP measures the precision of the ranked items for a query as the ratio of true rank and assigned rank:

$$
\text{MAP}(q) = \sum_{i=0}^{j-1} \frac{\text{rank of } q_i \text{ in } \{q'_0, \ldots, q'_{j-1}\} + 1}{i + 1},
$$

where the rank of $q_i$ is zero in case it does is not contained in the list $\{q'_0, \ldots, q'_{j-1}\}$ otherwise it is $i'$, s.t. $q_i = q'_{i'}$.

Our last metric called NDCG measures how the relevant substitutions are placed in the ranking list. It does not only compare the ranks of a substitution in the two rankings, but is also penalizes highly relevant substitutions according to $[q_0, \ldots, q_{j-1}]$ that have a very low rank in $[q'_0, \ldots, q'_{j-1}]$. Moreover, it takes the length of the actual lists into consideration. We refer the reader to the paper by Chakrabarti et al. [5] for details on NDCG.

The discussed metrics compare rankings for one query. To compare the utility of our algorithms, we...
average over all queries. For coverage we average over all queries for which the original search log produces substitutions. For all other metrics that try to capture the precision of a ranking, we average only over the queries for which the sanitized search logs produce substitutions. We generated query substitution only for the 100,000 most frequent queries of the original search log since the substitution algorithm only works well given enough information about a query.

In Figure 7 we vary $k$ and $\epsilon$ for $m = 1$ and we draw the utility curves for top-$j$ for $j = 2$ and $j = 5$. We observe that varying $\epsilon$ and $k$ has hardly any influence on the performance. On all precision measures, ZEALOUS provides utility comparable to $k$-query-anonymity. However, the coverage provided by ZEALOUS is not good. This is because the computation of query substitutions relies not only on the frequent query pairs but also on the count of phrase pairs which record for two sets of keywords how often a query containing the first set was followed by another query containing the second set. Thus a phrase pair can have a high frequency even though all query pairs it is contained in have very low frequency. ZEALOUS filters out these low frequency query pairs and thus loses many frequent phrase pairs.

As a last experiment, we study the effect of increasing $m$ for query substitutions. Figure 8 plots the average coverage of the top-2 and top-5 substitutions produced by ZEALOUS for $m = 1$ and $m = 6$ for various values of $\epsilon$. It is clear that across the board larger values of $m$ lead to smaller coverage, thus confirming our intuition in the previous section.

6 Related Work

The main focus of previous work on publishing search logs [1, 16, 19, 17] were different variants of $k$-anonymity [18]. Anonymity prevents re-identification of a user’s data in the published data; for search logs it means that the search history of every individual is indistinguishable from the history of $k$ − 1 other individuals.

Let us first discuss algorithms that have been suggested to achieve different types of of $k$-anonymity in search logs. Adar proposes the following algorithm: Given a search log partitioned into sessions, all queries are discarded that are associated with fewer than $k$ different user-ids. In each session the user-id is substituted by a random number [1]. We call the output a $k$-query anonymous search log. Motwani and Nabar substitute in each session the user-id by a random number and then add or delete Keywords from sessions until each session contains the same keywords as at least $k$ − 1 other sessions in the search log [16]. We call the output a $k$-session anonymous search log.

However, $k$-anonymity is insufficient for search logs since it does not prevent an attacker from learning sensitive information. We illustrate this through the following homogeneity attack [14] against a $k$-query anonymous search log. Similarly, a homogeneity attack against a $k$-session anonymous search log can be constructed.

Example 1 Suppose 100 different users only asked the query “prescription drugs under the counter SmallTown, XY”. This query is published in a 100-anonymous search log. It is not possible to link one of the occurrences of this query to a single user. But suppose only 100 of the 2000 inhabitants of SmallTown, XY have Internet access. Then an attacker concludes that each one of them shows the intention to buy prescription drugs illegally.
it is impossible for an attacker to link a search entry to its data-owner. Although $k$-query anonymity and $k$-session anonymity aim to achieve variants of $k$-anonymity, they fail to do so against active attacks as illustrated below.

**Example 2** An attacker wants to learn whether his neighbor (living at address $A_1$) who just moved into the town visited the cancer hospital (at address $A_2$). The attacker initiates $k - 1$ accounts and asks the query “from: $A_1$ to: $A_2$” with which her neighbor might try to calculate a route with a popular search engine. The attacker gets to see the $k$-anonymous search log. In case the query “from: $A_1$ to: $A_2$” appears in the sanitized search log the attacker concludes that her neighbor has the intention of visiting the local cancer hospital. In practice, an attacker might try different formulations of this query to cover all possible ways his neighbor could express this query.

This very simple and effective attack can be applied to a $k$-query-anonymous or $k$-session-anonymous search log. The attack shows that variations of $k$-anonymity do not actually prevent an attacker from linking sensitive information to an individual.

Apart from the algorithms that have been suggested for search logs there are more algorithms achieving variants of $k$-anonymity that could be applied to search logs. Multi-relational $k$-anonymity \cite{51} can be applied as a complement to $k$-query anonymity to publish clicks of a search log by encoding them as set-valued attributes. FreeForm-anonymity \cite{19} can be used to strengthen $k$-anonymity by considering more attributes as sensitive (not just the user-id). However, all these extensions are still vulnerable to the active attack which can be successfully carried out on the output of any anonymization algorithm that tries to achieve indistinguishability on the user-id level instead of the individual level.

Note that none of these attacks can be applied to a $(\epsilon, \delta)$-probabilistic differentially private search log. Even an attacker with multiple accounts learns roughly the same about a user whether or not the search history of that user was included in the published search log with high probability.

Independent of our work Korolova et al. \cite{12} developed a privacy-preserving algorithm for publishing queries and clicks.

### 7 Conclusions

In this paper we developed a novel algorithm called ZEALOUS that allows us to publish frequent keywords, queries, consecutive query pairs, and clicks. For many applications, infrequent items are of no interest, and thus our algorithm can provide good utility: The relative counts of the frequent items are preserved well as shown by our experiments.

However, there are applications for which the infrequent items matter. One such example is query

\footnote{At the writing of this paper, their work is not available yet, but we will compare their work with our algorithm once the paper becomes available.}
clustering [20], where the similarity measure of the clustering algorithm takes into account the implicit feedback given by user clicks. The actual count of a click does not matter to this clustering algorithm; it only matters whether or not the count is above zero. Since ZEALOUS only preserves the frequent clicks, ZEALOUS results in bad utility for this algorithm. Publishing information about infrequent items of a search log in a privacy-preserving manner remains a topic of future research.

Acknowledgments. We would like to thank our colleagues Filip Radlinski, Yisong Yue, and Jure Leskovec for helpful discussions about the usage of search logs.

References

[1] Eytan Adar. User 4xxxxx9: Anonymizing query logs. In WWW Workshop on Query Log Analysis, 2007.

[2] Roberto Baeza-Yates. Web usage mining in search engines. Web Mining: Applications and Techniques, 2004.

[3] Michael Barbaro and Tom Zeller. A face is exposed for aol searcher no. 4417749. New York Times http://www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000en=f6f61949c6da4d38ei=5090, 2006.

[4] Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In KDD, 2008.

[5] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In KDD, pages 88–96, 2008.

[6] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In OSDI, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[7] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In EUROCRYPT, 2006.

[8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In TCC, 2006.

[9] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. "I know what you did last summer": query logs and user privacy. In CIKM, 2007.

[10] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In WWW, 2006.

[11] Daniel Kifer and Johannes Gehrke. Injecting utility into anonymized datasets. In SIGMOD, 2006.

[12] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. To appear in WWW 2009.

[13] Ravi Kumar, Jasmine Novak, Bo Pang, and Andrew Tomkins. On anonymizing query logs via token-based hashing. In WWW, 2007.

[14] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In ICDE, 2006.

[15] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vihuber. Privacy: From theory to practice on the map. In ICDE, 2008.

[16] Rajeev Motwani and Shubha Nabar. Anonymizing unstructured data. arXiv, 2008.

[17] Mehmet Ercan Nergiz, Christopher Clifton, and Ahmet Erhan Nergiz. MultiRelational k-anonymity. IEEE Transactions on Knowledge and Data Engineering, 99(1), 2008.

[18] Pierangela Samarati. Protecting respondents’ identities in microdata release. IEEE Trans. on Knowl. and Data Eng., 13(6):1010–1027, 2001.
A Proof of Proposition 1

Let $\mathcal{A}$ denote the algorithm that adds noise to frequent keywords only. Let $SL$ be a search log containing search entries of $U$ users, and and $SL'$ be a neighboring search log of $SL$. Let $H$ ($H'$) be the keyword histogram constructed from $SL$ ($SL'$) by selecting $m$ distinct keywords from each user. Without loss of generality, assume that (i) $H$ contains exactly $U \cdot m/\tau$ keywords whose counts equal $\tau$, and (ii) any keyword in $H$ has a count $\tau - 1$ in $H'$. Observe that, given $SL'$, $\mathcal{A}$ always returns a histogram where none of the keywords in $H$ appears. On the other hand, given $SL$, with a non-zero probability $\mathcal{A}$ will return a histogram that contains some keywords in $H$. This clearly violates $\epsilon$-differential privacy.

Let $\Omega$ be the set of histograms that have zero counts for all keywords in $H$. If $\mathcal{A}$ achieves $(\epsilon, \delta)$-probabilistic differential privacy, then $\Pr[\mathcal{A}(SL) \in \Omega] \geq 1 - \delta$ must hold. But the following holds:

$$
\Pr[\mathcal{A}(SL) \in \Omega] = \Pr\left[\text{All keywords in } H \text{ have zero counts in } \mathcal{A}(SL)\right] = \left(\int_{-\infty}^{-\tau} \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} dx \right)^{U \cdot m/\tau} = \left(\frac{1}{2} e^{-\frac{x}{\lambda}}\right)^{U \cdot m/\tau}.
$$

This indicates that $\Pr[\mathcal{A}(SL) \in \Omega] \geq 1 - \delta$ does not hold for any $\delta < 1/2$, completing our proof.