BLENDER: Enabling Local Search with a Hybrid Differential Privacy Model

Brendan Avent
University of Southern California
bavent@usc.edu

Aleksandra Korolova
University of Southern California
korolova@usc.edu

David Zeber
Mozilla
dzeber@mozilla.com

Torgeir Hovden
Mozilla
torgeir@eritreum.com

Benjamin Livshits
Imperial College London
livshits@ic.ac.uk

Abstract

We propose a *hybrid* model of differential privacy that considers a combination of regular and opt-in users who desire the differential privacy guarantees of the local privacy model and the trusted curator model, respectively. We demonstrate that within this model, it is possible to design a new type of *blended* algorithm for the task of privately computing the head of a search log. This blended approach provides significant improvements in the utility of obtained data compared to related work while providing users with their desired privacy guarantees. Specifically, on two large search click data sets, comprising 1.75 and 16 GB respectively, our approach attains NDCG values exceeding 95% across a range of privacy budget values.

1 Introduction

Now more than ever we are confronted with the tension between collecting mass-scale user data and the ability to release or share this data in a way that preserves the privacy of individual users. Today, an organization that needs user data to improve the quality of service they provide often has no choice but to perform the data collection themselves. However, the users may not want to share their data with the organization, especially if they consider the data to be sensitive or private. Similarly, the organization assumes liability by collecting sensitive user data: private information may be directly leaked through security breaches or subpoenas, or indirectly leaked by the output of computations done on the data. Thus, both organizations and users would benefit not only from strong, rigorous privacy guarantees regarding the data collection process, but also from the organization collecting the minimum amount of data necessary to achieve their goal. Some of the philosophy behind our work stems from a desire to enable privacy-preserving *decentralized* data collection that aggregates data from multiple entities into high quality datasets.

1.1 Differential Privacy and Curator Models

In the last decade, we have witnessed scores of ad-hoc approaches that have turned out to be inadequate for protecting privacy [33, 23]. The problem stems from the impossibility of foreseeing all attacks of adversaries capable of utilizing outside knowledge. Differential privacy [10, 9, 11], which has become the gold standard privacy guarantee in the academic literature, and is gaining traction in industry and government [13, 17, 28], overcomes the prior issues by focusing on the *privatization* algorithm applied to the data, requiring that it preserves privacy in a mathematically rigorous sense under an assumption of an omnipotent adversary.

There are two primary models in the differential privacy framework that define how data is to be handled by the users and data collectors: the *trusted curator* model and the *local* model.

Trusted curator model: Most differentially private algorithms developed to date operate in the *trusted curator* model: all users' data is collected by the curator before privatization techniques are applied to it. In this model, although users are guaranteed that the released data set protects their privacy, they must be willing to share their private, unperturbed data with the curator and trust that the curator properly performs a privacy-preserving perturbation.

Local model: As was most recently argued by Apple [17], users may not trust the data collector with their data, and may prefer privatization to occur before their data reaches the collector. Since privatization occurs locally, this is known as the *local differential privacy* (LDP) model, or *local* model. Over the last several years, we have seen some examples

of the local model beginning to be used for data collection in practice, most notably in the context of the Chrome web browser [13] and Apple's data collection [17].

In the LDP model, a data collector such as Google or Apple obtains insights into the data without observing the exact values of user's private data. This is achieved by applying a privacy-preserving perturbation to each user's private data before it leaves the user's device. Since most people do not trust web companies with maintaining the privacy and security of their data [29], the minimal trust required of users towards the data collector is a very attractive property of the LDP model. This approach protects not only the individual users, but also the data collector from the possible privacy breaches. For these reasons, the local model directly embodies the "data minimization" principle described in the White House's 2012 consumer data privacy report [40].

Although it may seem counter-intuitive, it is possible to obtain useful insights even when the data collector does not have access to the original data and receives only data that has already been locally privatized. Suppose a data collector wants to determine the proportion of the population that is HIV-positive. The local privatization algorithm works as follows: each person contributing data secretly flips a coin. If the coin lands heads, they report their true HIV status; otherwise, they report a status at random. This algorithm, known as *randomized response* [39], guarantees each person plausible deniability and is differentially private. Since the randomness is incorporated into the algorithm in a precisely specified way, the data collector is able to recover an accurate estimate of the true proportion of HIV-positive people if enough people contribute their locally privatized data.

Differential privacy: Formally, an algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private [11] if and only if for all neighboring databases D and D' differing in precisely one user's data, the following inequality is satisfied for all possible sets of outputs $Y \subseteq Range(\mathcal{A})$:

$$\Pr[\mathcal{A}(D) \in Y] \leq e^{\epsilon} \Pr[\mathcal{A}(D') \in Y] + \delta.$$

The definition of what it means for an algorithm to preserve differential privacy is the same for both the trusted curator model and the local model. The only distinction is in the timing of when the privacy perturbation needs to be applied – in the local model, the data needs to undergo a privacy-preserving perturbation before it is sent to the aggregator, whereas in the trusted curator model the aggregator may first collect all the data, and then apply a privacy-preserving perturbation. The timing distinction leads to differences in what is meant by "neighboring databases" in the definition and which algorithms are analyzed. In the local model, D represents data of a single user and D' represents data of the same user, with possibly changed values. In the trusted curator model, D represents data of all users and D' represents data of all users, except values of one of the user's data may be altered.

Current differential privacy literature considers the trusted curator model and the local model entirely independently. Our goal is to show that there is much to be gained by combining the two.

Hybrid model: Much of the contribution in this paper stems from our observation that the two models can co-exist. As others have observed [2, 1, 7], people's attitudes toward privacy vary widely. Specifically, some users may be comfortable with sharing their data with a trusted curator.

Many companies rely on a group of beta testers with whom they have higher levels of mutual trust. It is not uncommon for such beta testers to voluntarily opt-in to a less privacy-preserving model than that of an average end-user [32]. For example, Mozilla warns potential beta users of its Firefox browser that "Pre-release versions automatically send Telemetry data to Mozilla to help us improve Firefox[1]"; Google has a similar provision for the beta testers of the Canary build of the Chrome browser[2].

For the users who have higher trust in the company — we call them the *opt-in group*, the trusted curator privacy model is a natural match. For all other users — we call them *clients*, the local privacy model is appropriate. Our goal is to demonstrate that by separating the user pool into these two groups, according to their trust (or lack thereof) in the data aggregator, we can improve the utility of the collected data. We dub this new model the *hybrid differential privacy* model.

1.2 Applications

Heavy hitter discovery and estimation is a well-studied problem in the context of information retrieval, and is one of the canonical problems in privacy-preserving data analysis [6, 27]. Moreover, recent work in the LDP model is focused on precisely that problem [13, 34] or very closely related ones of histogram computations [5, 21]. However, current privacy-preserving approaches in the LDP

[1] https://www.mozilla.org/en-US/privacy/firefox/

[2] https://www.chromium.org/getting-involved/dev-channel

model lead to utility losses that are quite significant, sometimes to the point where results are no longer usable. Clearly, if the privacy-preserving perturbation makes the data deviate too far from the original, the approach will not be widely adopted. This is especially true in the context of search tasks, where users have been conditioned for years to expect high-quality results.

We consider two specific applications in the space of heavy hitter estimation: local search and search trend computation.

Local search: Much of the work in this paper is motivated by the idea of *local search*, an application of heavy hitter estimation. Local search revolves around the problem of how a browser maker can collect information about users' clicks as they interact with search engines in order to create the *head* of the search, i.e., the collection of the most popular queries and their corresponding URLs. Specifically, it involves computing on query-URL pairs, where the URLs are those clicked as a result of submitting the query and receiving a set of answers.

Note that local search results may be computed by combining the results obtained from user interactions that stem from several search engines, depending on the context. For instance, the user's current geographic location may lead the browser maker to surface results obtained from Baidu and not Bing.

With proper privacy measures in place, this data set can be deployed in the end-user browser to serve the most common queries with a very low latency or in situations when the user is disconnected from the network. Local search can be thought of as a form of caching, where many queries are answered in a manner that does not require a round trip to the server. Such caching of the most frequently used queries locally has a disproportionately positive impact on the expected query latency [36, 3] as queries to a search engine follow a power-law distribution [4]. Furthermore, it would not be unusual or require a significantly novel infrastructure, as plenty of data is delivered to the browser today, such as SafeBrowsing malware databases in Chrome and Firefox, Microsoft SmartScreen data in Internet Explorer, blocking lists for extensions such as AdBlock Plus, etc.

Trend computation: Search trend computation is a typical example of heavy hitter estimation. This problem entails finding the most popular queries and sorting them in order of popularity; think about it as the top-10 computation based on local search observations. An example of this is the Google trends service[3], which has an always up-to-date list of trending topics and queries.

Although trend computation is interesting, local search is a great deal harder to do well on while preserving most of the utility. Luckily, in the domain of search quality, there are established metrics to numerically assess the quality of search results; one of such metrics is NDCG, and we rely on it heavily in assessing the performance of our proposed system.

1.3 Contributions

Our paper makes the following contributions:

- We introduce and utilize a more realistic, hybrid trust model, which removes the need for all users to trust a central curator.
- We propose Blender, an algorithm that operates with the hybrid differential privacy model for computing heavy hitters. Blender blends the data of opt-in and all other users in order to improve the resulting utility.
- We test Blender on two common applications: *search trend computation* and *local search* and find that it preserves high levels of utility while maintaining differential privacy for reasonable privacy parameter values.
- As part of Blender, we propose an approach for automatically balancing the data obtained from participation of opt-in users with that of other users to maximize the eventual utility.
- We perform a comprehensive utility evaluation of Blender on two large search click data sets, comprising 1.75 and 16 GB, demonstrating that Blender maintains very high level of utility (i.e., NDCG values in excess of 95% across a range of parameters).

2 System Overview

We now discuss the high-level overview of our proposed system, Blender, that coordinates the privatization, collection, and aggregation of data in the hybrid model, as well as some of the specific choices we make in this system. We use the task of enabling local search based on user histories while preserving differential privacy throughout, but, as will become clear from the discussion, our model and system can also be applied to other frequency-based estimation tasks. As discussed in Section 1, we consider two groups of users: the opt-in group, who are comfortable with privacy as ensured by the trusted curator model, and the clients, who desire the privacy guarantees of the local model.

[3] https://www.google.com/trends/

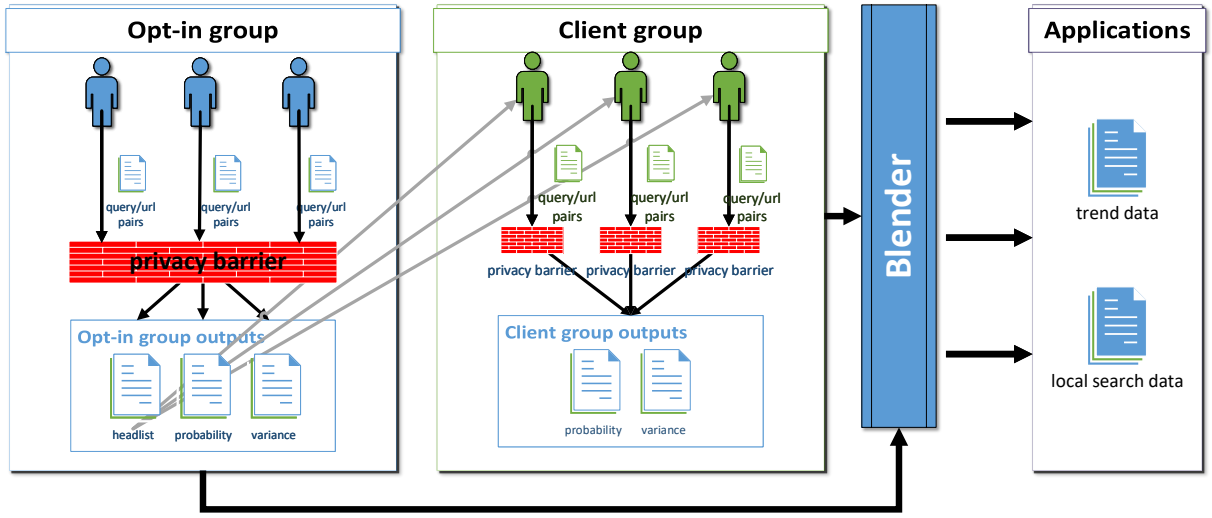


Figure 1: Architectural diagram of Blender's processing steps.

2.1 Outline of Our Approach

The core of our innovation is to take advantage of the privatized information obtained from the opt-in group in order to create a more efficient (in terms of utility) algorithm for data collection from the clients. Furthermore, the privatized results obtained from the opt-in group and from the clients are then "blended" in a way that takes into account the privatization algorithms used for each group, and thus, again, achieving an improved utility over a less-informed combination of data from the two groups.

The problem of enabling privacy-preserving local search using past search histories can be viewed as the task of identifying the most frequent search records among the population of users, and estimating their underlying probabilities (both in a differential privacy-preserving manner). In this context, we call the data collected from the users *search records*, where each search record is a pair of strings of the form $\langle query, URL \rangle$, representing a query that a user posed followed by the URL that the user subsequently clicked. We denote by $p_{\langle q,u \rangle}$ the true underlying probability of the search record $\langle q, u \rangle$ in the population. We assume that our system receives a sample of users from the population, each holding their own collection of private data drawn independently and identically from the distribution over all records p. Its goal is to output an estimate $\hat{p}$ of probabilities of the most frequent search records, while preserving differential privacy (in the trusted curator model) for the opt-in users and (in the local model) for the clients.

Informal Overview of Blender: Figure 1 presents an architectural diagram of Blender.

Blender serves as the trusted curator for the opt-in group of users, and begins by aggregating data from them. Using a portion of the data, it constructs a candidate *head list* of records in a differentially private manner that approximates the most common search records in the population. It additionally includes a single "wildcard" record, $\langle \star, \star \rangle$, which represents all records in the population that weren't previously included in the candidate head list. It then uses the remainder of the opt-in data to estimate the probability of each record in the candidate head list in a differentially private manner, and (optionally) trims the candidate head list down to create the final head list. The result of this component of Blender is the privatized trimmed head list of search records and their corresponding probability and variance estimates, which can be shared with each user in the client group, and with the world.

Each member of the client group receives the privatized head list obtained from the opt-in group. Each client then uses the head list to apply a differential privacy-preserving perturbation to their data, subsequently reporting their perturbed results to Blender. Blender then aggregates all the clients' reports and, using a statistical denoising procedure, estimates both the probability for each record in the head list as well as the variance of each of the estimated probabilities based on the clients' data.

For each record, Blender combines the record's probability estimates obtained from the two groups. It does so by taking a convex combination of the

groups' probability estimates for each record, carefully weighted based on the record's variance estimate in each group. The combined result under this weighting scheme yields a better probability estimate than either group is able to achieve individually. Finally, BLENDER outputs the obtained records and their combined probability estimates, which can be used to drive local search, determine trends, etc.

A Formal Overview of Blender: Figure 2 presents the precise algorithmic overview of each step, including key parameters. Lines 1-3 of BLENDER describe the treatment of data from opt-in users, line 4 – the treatment of clients, and line 5 – the process for combining the probability estimates obtained from the two groups. The only distinction between opt-in users and clients in terms of privacy guarantees provided is the curator model – trusted curator and local model, respectively. Other than that, both types of users are assumed to desire the same level of (ϵ, δ)-differential privacy.

We will detail our choices for the privatization sub-algorithms and discuss their privacy properties next. A key feature of BLENDER, however, is that its privacy properties do not depend on the specific choices of the sub-algorithms. That is, as long as CREATEHEADLIST, ESTIMATEOPTINPROBABILITIES, and ESTIMATECLIENTPROBABILITIES each satisfy (ϵ, δ)-differential privacy in its respective curator model, then so does BLENDER. This allows changing the sub-algorithms if better versions (utility-wise or implementation-wise) are discovered in the future. Among the parameters of BLENDER, the first four (the privacy parameters and the sets of opt-in and client users) can be viewed as given externally, whereas the following five (the number of records collected from each user and the distribution of the privacy budget among the sub-algorithms' sub-components) can be viewed as knobs the designer of BLENDER is at liberty to tweak in order to improve the overall utility of BLENDER's results.

BLENDER $(\epsilon, \delta, O, C, m_O, m_C, f_O, f_C, M)$

Parameters:

- ϵ, δ: the differential privacy parameters.
- O, C: the set of opt-in users and clients, respectively.
- m_O, m_C: the max number of records to collect from each opt-in / client user, respectively.
- f_O: the fraction of the opt-in users to use in head list creation (the remainder are used to estimate the record probabilities).
- f_C: the fraction of the clients' privacy budget to allocate to queries (as opposed to URLs).
- M: the maximum size of the finalized head list.

Variables:

- HL_S, HL: a map from each query to its corresponding set of URLs.
- $\hat{p}_O, \hat{\sigma}^2_O, \hat{p}_C, \hat{\sigma}^2_C$: vectors indexed by records in HL (and, overloaded to be indexed by queries in HL as well) containing the probability estimates and variance estimates for each record (and query).

Body

1: Arbitrarily partition O into S and $T = O \setminus S$, such that $|S| = f_O|O|$ and $|T| = (1 - f_O)|O|$.
2: **let** HL_S = CREATEHEADLIST$(\epsilon, \delta, S, m_O)$ be the initial head list of records computed based on opt-in users' data.
3: **let** $\langle HL, \hat{p}_O, \hat{\sigma}^2_O \rangle$ = ESTIMATEOPTINPROBABILITIES$(\epsilon, \delta, T, m_O, HL_S, M)$ be the refined head list of records, their estimated probabilities, and estimated variances based on opt-in users' data.
4: **let** $\langle \hat{p}_C, \hat{\sigma}^2_C \rangle$ = ESTIMATECLIENTPROBABILITIES$(\epsilon, \delta, C, m_C, f_C, HL)$ be the estimated record probabilities and estimated variances based on client reports.
5: **let** $\hat{p}$ = BLENDPROBABILITIES$(\hat{p}_O, \hat{\sigma}^2_O, \hat{p}_C, \hat{\sigma}^2_C, HL)$ be the combined estimate of record probabilities.
6: **return** $HL, \hat{p}$.

Figure 2: BLENDER, the server algorithm that coordinates the privatization, collection, and aggregation of data from all users.

2.2 Overview of Blender Sub-Algorithms

We now present the specific choices we made for the sub-algorithms in BLENDER. Detailed technical discussions of their properties follow in Section 3.

Algorithms for Head List Creation and Probability Estimation Based on Opt-in User Data (Figures 3, 4): The opt-in users are partitioned into two sets – S, whose data will be used for initial head list creation, and T, whose data will be used to estimate the probabilities and variances of records from the formed initial head list.

The initial head list creation algorithm, described in Figure 3, constructs the list in a differentially private manner using search record data from group S. The goal of the algorithm is to approximate the true set of most frequently searched and clicked search records as closely as possible, while ensuring differential privacy. The algorithm follows the strategy introduced in [26] by aggregating the records of the opt-in users from S, and including in the head list those records whose noisy count exceeds a threshold. The noise to add to the true counts[4] and the threshold to use are calibrated to ensure differential privacy, using [24].

Our algorithm differs from previous work in two

[4]Whenever we refer to noise as Lap(b), we mean a random draw from the Laplace distribution with scale b.

CreateHeadList(ϵ, δ, S, m_O)

Parameters:

- ϵ, δ: the differential privacy parameters.
- S: a set of opt-in users.
- m_O: the maximum number of records to collect from each opt-in user.

Body

1: **let** $N(r, D)$ = number of times an arbitrary record r appears in the given dataset D.
2: **for** each user $i \in S$ **do**
3: **let** $D_{S,i}$ be the database aggregating at most m_O arbitrary records from i.
4: **let** D_S be the concatenation of all $D_{S,i}$ databases.
5: **let** HL_S be an empty map.
6: $b_S = \frac{2m_O}{\epsilon}$.
7: $\tau = b_s \cdot \big(\ln(\exp(\frac{\epsilon}{2}) + m_O - 1) - \ln(\delta)\big)$.
8: Assert $\tau \geq 1$.
9: **for** each distinct $\langle q, u\rangle \in D_S$ **do**
10: **let** Y be an independent draw from Lap(b_S).
11: **if** $N(\langle q, u\rangle, D_S) + Y > \tau$ **then**
12: Add q to HL_S if $q \notin HL_S$.
13: Append u to $HL_S[q]$.
14: Add $\langle\star, \star\rangle$ to HL_S.
15: **return** HL_S.

Figure 3: Algorithm for creating the head list from a portion opt-in users in a privacy-preserving way.

ways: 1) it replaces the collection and thresholding of queries with the collection and thresholding of records (i.e., query - URL pairs) and 2) its definition of neighboring databases is that of databases differing in one user's record values, rather than in the removal or addition of one user's data. These distinctions necessitate the choice of $m_O = 1$, as well as higher values for noise and threshold than in [24].

We introduce a wildcard record $\langle\star, \star\rangle$ to represent records not included in the head list, for the subsequent task of estimating their aggregate probability.

For each record included in the initial head list, the algorithm described in Figure 4 uses the remaining opt-in users' data (from set T) to differentially privately estimate their probabilities, denoted by $\hat{p}_O$. This algorithm is the standard Laplace mechanism from the differential privacy literature [10], with scale of noise calibrated to output sensitivity due to our definition of neighboring datasets. Our implementation ensures $(\epsilon, 0)$-differential privacy, which is a more stringent privacy guarantee than for any non-zero δ. We need to set $m_O = 1$ for the privacy guarantees to hold, because we treat data at the search record rather than query level.

We form the final head list from the M most frequent records in $\hat{p}_O$. Finally, the head list is passed to the client group, and the head list and its probability and variance estimates are passed to the BlendProbabilities step of Blender.

EstimateOptinProbabilities($\epsilon, \delta, T, m_O, HL_S, M$)

Parameters:

- ϵ, δ: the differential privacy parameters. In fact, this algorithm achieves $(\epsilon, 0)$-differential privacy, which is a stricter privacy guarantee than (ϵ, δ)-differential privacy, for all $\delta > 0$.
- T: a set of opt-in users.
- m_O: the maximum number of records to collect from each opt-in user.
- HL_S: the initial head list of records whose probabilities are to be estimated.
- M: the maximum size of the finalized head list.

Body

1: **let** $N(r, D)$ = number of times an arbitrary record r appears in the given dataset D.
2: **for** each user $i \in T$ **do**
3: **let** $D_{T,i}$ be the database aggregating at most m_O arbitrary records from i.
4: **let** D_T be the concatenation of all $D_{T,i}$ databases.
5: Transform any record $\langle q, u\rangle \in D_T$ that doesn't appear in HL_S into $\langle\star, \star\rangle$.
6: **let** $\hat{p}_O$ be a vector indexed by records in HL_S containing the respective probability estimates.
7: **let** $\hat{\sigma}^2_O$ be a vector indexed by records in HL_S containing variance estimates of the respective probability estimate.
8: Denote $|D_T|$ as the total number of records in D_T.
9: **let** $b_T = \frac{2m_O}{\epsilon}$.
10: **for** each $\langle q, u\rangle \in HL_S$ **do**
11: **let** Y be an independent draw from Lap(b_T).
12: $\hat{p}_{O,\langle q,u\rangle} = \frac{1}{|D_T|}(N(\langle q, u\rangle, D_T) + Y)$.
13: $\hat{\sigma}^2_{O,\langle q,u\rangle} = \frac{\hat{p}_{O,\langle q,u\rangle}(1-\hat{p}_{O,\langle q,u\rangle})}{|D_T|-1} + \frac{2b_T^2}{|D_T|\cdot(|D_T|-1)}$.
14: **let** HL map the M queries with the highest estimated marginal probabilities (according to $\hat{p}_O$) to their respective sets of URLs.
15: For the records not retained in HL, accumulate their estimated probabilities into $\hat{p}_{O,\langle\star,\star\rangle}$ and update $\hat{\sigma}^2_{O,\langle\star,\star\rangle}$ as in line 13.
16: **return** $HL, \hat{p}_O, \hat{\sigma}^2_O$.

Figure 4: Algorithm for privacy-preserving estimation of probabilities of records in the head list from a portion of opt-in users.

The choice of how to split opt-in users into the sub-groups of S and T and the choice of M are unrelated to privacy constraints, and can be made by Blender's developer to optimize utility goals, as will be discussed in Section 4.2.1.

The technical discussions of the algorithms' privacy properties and variance estimate computations follow in Section 3.1 and Section 3.3.

Algorithms for client data collection (Figures 5, 6): For privatization of client data, the records are no longer treated as a single entity, but rather in a two-stage process: first privatizing the

EstimateClientProbabilities$(\epsilon, \delta, C, m_C, f_C, HL)$

Parameters:

- ϵ, δ: the differential privacy parameters.
- C: the set of clients.
- m_C: the number of records to collect from the client.
- f_C: the fraction of the privacy budget to allocate to reporting queries.
- HL: a map from each query to its corresponding set of URLs.

Body

Append query $q = \star$ to HL.
for each query $q \in HL$ **do**
    Append URL $u = \star$ to $HL[q]$.
for each client $i \in C$ **do**
    let $D_{C,i} = \text{LocalAlg}(\epsilon, \delta, m_C, f_C, HL)$ be the reports from i's local execution of LocalAlg.
let D_C be the concatenation of all reported client datasets, $D_{C,i}$.
Denote $|D_C|$ as the total number of records in D_C.
let variables $\epsilon'_Q, \epsilon'_U, \delta'_Q, \delta'_U, k, t, k_q, t_q (\forall q \in HL)$ be defined as in lines 2–4 of LocalAlg.
let $\hat{r}_C, \hat{p}_C, \hat{\sigma}^2_C$ be vectors indexed by records in HL (and overloading its use, also indexed by queries).
for $q \in HL$ **do**
    let $\hat{r}_{C,q}$ be the fraction of queries q in D_C.
    $\hat{p}_{C,q} = \frac{\hat{r}_{C,q} - \frac{1-t}{k-1}}{t - \frac{1-t}{k-1}}$
    $\hat{\sigma}^2_{C,q} = \frac{1}{\left(t - \frac{1-t}{k-1}\right)^2} \frac{\hat{r}_{C,q}(1-\hat{r}_{C,q})}{|D_C|-1}$
    for $u \in HL[q]$ **do**
        let $\hat{r}_{C,\langle q,u\rangle}$ be the fraction of records which are $\langle q, u\rangle$ in D_C.
        $\hat{p}_{C,\langle q,u\rangle} = \frac{\hat{r}_{C,\langle q,u\rangle} - \frac{(1-t_q)t\hat{p}_{C,q}}{k_q - 1} - \frac{(1-t)(1-\hat{p}_{C,q})}{(k-1)k_q}}{t(t_q - \frac{1-t_q}{k_q - 1})}$
        $\hat{\sigma}^2_{C,\langle q,u\rangle} = \left(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|-1} + \frac{2|D_C|}{|D_C|-1}\left(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\right)\left(\frac{k-2+t}{kt-1}\right)\hat{r}_{C,\langle q,u\rangle} + \left(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\right)^2 \hat{\sigma}^2_{C,q}\right) \cdot \frac{1}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2}$
return $\hat{p}_C, \hat{\sigma}^2_C$.

Figure 5: Algorithm for estimating probabilities of records in the head list from the locally privatized reports of the client users.

LocalAlg$(\epsilon, \delta, m_C, f_C, HL)$

Parameters:

- ϵ, δ: the differential privacy parameters.
- m_C: the number of records to collect from the client.
- f_C: the fraction of the privacy budget to allocate to reporting queries.
- HL: the head list, represented as a map keyed by queries $\{q_1, \ldots, q_k, \star\}$. The value for each $q \in HL$ is defined as $HL[q] = \{u_1, \ldots, u_l, \star\}$, representing all URLs in the head list associated with q.

Body

1: **let** $D_{C,i}$ be the database aggregating at most m_C records from current client i.
2: $\epsilon' = \epsilon/m_C$, and $\delta' = \delta/m_C$.
3: $\epsilon'_Q = f_C\epsilon', \epsilon'_U = \epsilon' - \epsilon'_Q$ and $\delta'_Q = f_C\delta', \delta'_U = \delta' - \delta'_Q$.
4: $k = |HL|$, and $t = \frac{\exp(\epsilon'_Q) + (\delta'_Q/2)(k-1)}{\exp(\epsilon'_Q) + k - 1}$.
5: **for** each $q \in HL$ **do**:
6: $k_q = |HL[q]|$, and $t_q = \frac{\exp(\epsilon'_U) + (\delta'_U/2)(k_q - 1)}{\exp(\epsilon'_U) + k_q - 1}$.
7: **for** each $\langle q, u\rangle \in D_{C,i}$ **do**
8: **if** $q \notin HL$ **then**
9: Set $q = \star$.
10: **if** $u \notin HL[q]$ **then**
11: Set $u = \star$.
12: With probability $(1 - t)$,
13: **let** q' be a unif. random query from $HL \setminus q$.
14: **let** u' be a unif. random URL from $HL[q']$.
15: **report** $\langle q', u'\rangle$.
16: **continue**
17: With probability $(1 - t_q)$,
18: **let** u' be a unif. random URL from $HL[q] \setminus u$.
19: **report** $\langle q, u'\rangle$.
20: **continue**
21: **report** $\langle q, u\rangle$.

Figure 6: Algorithm executed by each client for privately reporting their records.

query, then privatizing the URL. This choice is intended to benefit utility as the number of queries is significantly larger than the number of URLs associated with any query, and hence allocating a larger portion of the privacy budget to the query-reporting stage is a prudent choice.

The process of local privatization of each client's value (Figure 6) follows the strategy of the Exponential mechanism introduced by [30]. The privatization algorithm reports the true value with a certain bounded probability, and otherwise, randomizes the answer uniformly among all the other feasible values.

The fact that the head list (approximating the set of the most frequent records) is available to each client plays a crucial role in improving the utility of the data produced by this privatization algorithm compared to the previously known algorithms operating in the local privacy model. Knowledge of the head list allows dedicating the entire privacy budget to report the true value, rather than having to allocate some of it for estimating an analogue of the head list, as done in [15, 34]. Another distinction from the Exponential mechanism designed to improve utility is utilization of δ.

The choices of m_C and f_C are not related to privacy constraints, and can be made by Blender's developer to optimize utility goals, as will be discussed in Section 4.2.1.

The local nature of the privatization algorith, i.e., the use of a randomization procedure that can report any record with some probability, induces a predictable bias to the distribution of reported records. The removal of this bias, which we refer to as *denoising* (discussed further in Section 3.2), results in the proper probability estimates $\hat{p}_C$ (Figure 5). These probability estimates along with the variance estimates are then passed to the BLENDPROBABILITIES part of BLENDER.

The technical discussion of the algorithm's privacy properties, the denoising procedure and variance estimate computations follow in Sections 3.2 and 3.3.

Algorithm for Blending (Figure 7): The blending portion of BLENDER combines the estimates produced by the opt-in and client probability-estimation algorithms by taking into account the sizes of the groups and the amount of noise each sub-algorithm added. This produces a blended probability estimate $\hat{p}$ which, in expectation, is more accurate than either group produced individually. The procedure for blending is not subject to privacy constraints, as it operates on the data whose privacy has already been ensured by previous steps of BLENDER. The motivation and technical discussion of blending follows in Section 3.3.

BLENDPROBABILITIES($\hat{p}_O, \hat{\sigma}^2_O, \hat{p}_C, \hat{\sigma}^2_C, HL$)

Parameters:

- $\hat{p}_O, \hat{p}_C$: the probability estimates from the opt-in and client algorithms.
- $\hat{\sigma}_O, \hat{\sigma}_C$: the variance estimates from the opt-in and client algorithms.
- HL: the head list of records.

Body

1: **let** $\hat{p}$ be a vector indexed by records in HL.
2: **for** $\langle q, u\rangle \in HL$ **do**
3: $\quad w_{\langle q,u\rangle} = \frac{\hat{\sigma}^2_{C,\langle q,u\rangle}}{\hat{\sigma}^2_{O,\langle q,u\rangle} + \hat{\sigma}^2_{C,\langle q,u\rangle}}$.
4: $\quad \hat{p}_{\langle q,u\rangle} = w_{\langle q,u\rangle} \cdot \hat{p}_{O,\langle q,u\rangle} + (1 - w_{\langle q,u\rangle}) \cdot \hat{p}_{C,\langle q,u\rangle}$.
5: **Optional:** Project $\hat{p}$ onto probability simplex (e.g., see [38]).
6: **return** $\hat{p}$.

Figure 7: Algorithm for combining record probability estimates from opt-in and client estimates.

3 Technical Detail Summary

We now present further technical details related to the instantiations of the sub-algorithms for BLENDER, such as statements of privacy properties and the motivation for BLENDPROBABILITIES.

3.1 Opt-in Data Algorithms

Differential privacy of the algorithms handling opt-in client data follows directly from previous work.

Theorem 1. *([24])* CREATEHEADLIST *guarantees* (ϵ, δ)*-differential privacy if* $m_O = 1, \epsilon > \ln(2)$, *and* $\tau \geq 1$.

Theorem 2. *([10])* ESTIMATEOPTINPROBABILITIES *guarantees* $(\epsilon, 0)$*-differential privacy if* $m_O = 1$.

3.2 Client Data Algorithms

LOCALALG is responsible for the privacy-preserving perturbation of each client's data before it gets sent to the server, and ESTIMATECLIENTPROBABILITIES is responsible for aggregating the received privatized data into a meaningful statistic. We present the privacy statement and explain the logic behind the aggregation procedure next and prove them in Appendix A.

Theorem 3. LOCALALG *is* (ϵ, δ)*-differentially private.*

Denoising: The reports aggregated by the client mechanism form an empirical distribution over the records (and queries). Relative to the true underlying record distribution, this distribution is biased in an explicit and publicly-known way, as described by the reporting process. Thus, we seek to obtain an unbiased estimate of the true record distribution from this reported distribution. Concretely, we refer to this as *denoising* the reported empirical distribution $\hat{r}_C$ to obtain the final estimate from the client algorithm, $\hat{p}_C$. The denoising procedure relies only on the publicly-known reporting process as well as the already-privatized reports. Thus, this can be considered a *post-processing* step, which has no negative impact on the differential privacy guarantee [11] yet significantly improves utility.

Observation 1. $\hat{p}_C$ *gives the unbiased estimate of record and query probabilities under* ESTIMATECLIENTPROBABILITIES.

3.3 Blending

The opt-in algorithm and the client algorithm both output independent estimates $\hat{p}_O$ and $\hat{p}_C$ of the record distribution p. The question we address now is how to best combine these estimates using the information available.

A standard way to measure the quality of an estimate is by its variance. Although it may seem natural to choose the estimate with lower variance as the final estimate $\hat{p}$, it is possible to achieve a better estimate by jointly utilizing the information provided by

both algorithms. This is because the errors in these algorithms' estimates come from different, independent sources. The error in the estimates obtained from the opt-in algorithm is due to the addition of noise, whereas the error in the estimates obtained from the client algorithm is due to randomization of the reports over the set of records in the head list. Thus, if we determine the variances of the estimates obtained from the two algorithms, we can use these variances to *blend* the estimates in the best way.

More formally, for each record $\langle q, u\rangle$ let $\sigma^2_{O,\langle q,u\rangle}$ and $\sigma^2_{C,\langle q,u\rangle}$ be the variances of the opt-in and client algorithm's estimates of $\hat{p}_{O,\langle q,u\rangle}$ and $\hat{p}_{C,\langle q,u\rangle}$ respectively. Since these variances depend on the underlying distribution, which is unknown a priori, we will compute sample variances $\hat{\sigma}^2_{O,\langle q,u\rangle}$ and $\hat{\sigma}^2_{O,\langle q,u\rangle}$ instead. For each record $\langle q, u\rangle$, we will weigh the estimate from the opt-algorithm by $w_{\langle q,u\rangle}$ and the estimate from the client algorithm by $(1-w_{\langle q,u\rangle})$, where $w_{\langle q,u\rangle}$ is defined as in line 3 of BLENDPROBABILITIES. The optional step of projecting the blended estimates (e.g., as in [38]) ensures that the estimates sum to 1 and are non-negative.

Theorem 4 presents our computation of the sample variance of ESTIMATEOPTINPROBABILITIES, Theorem 5 presents our computation of the sample variance of ESTIMATECLIENTPROBABILITIES, and Theorem 6 motivates the weighting scheme used in BLENDPROBABILITIES. Their proofs are presented in Appendix B.

For the variance derivations, we make an explicit assumption that each piece of reported data is drawn independently and identically from the same underlying distribution. This is reasonable when comparing data across users. By setting $m_O = m_C = 1$, we remove the need to assume iid data *within* each user's own data, while simplifying our variance computations. We show in Section 4 that BLENDER achieves high utility even when $m_O = m_C = 1$.

Theorem 4. *When $m_O = 1$ the unbiased variance estimate for* ESTIMATEOPTINPROBABILITIES *can be computed as:* $\hat{\sigma}^2_{O,\langle q,u\rangle} = \frac{|D_T|}{|D_T|-1}\left(\frac{\hat{p}_{O,\langle q,u\rangle}(1-\hat{p}_{O,\langle q,u\rangle})}{|D_T|} + 2\left(\frac{b_T}{|D_T|}\right)^2\right)$.

Theorem 5. *When $m_C = 1$ the unbiased variance estimate for* ESTIMATECLIENTPROBABILITIES *can be computed as:* $\hat{\sigma}^2_{C,\langle q,u\rangle} = \frac{1}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2}\cdot\left(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|-1} + \left(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\right)^2\hat{\sigma}^2_{C,q} + \frac{2|D_C|}{|D_C|-1}\left(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\right)\left(\frac{k-2+t}{kt-1}\right)\hat{r}_{C,\langle q,u\rangle}\right)$.

Theorem 6 (Sample Variance Optimal Weighting)**.** *If $\hat{\sigma}^2_{O,\langle q,u\rangle}$ and $\hat{\sigma}^2_{C,\langle q,u\rangle}$ are sample variances of $\hat{p}_{O,\langle q,u\rangle}$ and $\hat{p}_{C,\langle q,u\rangle}$ respectively, then $w_{\langle q,u\rangle} = \frac{\hat{\sigma}^2_{C,\langle q,u\rangle}}{\hat{\sigma}^2_{O,\langle q,u\rangle}+\hat{\sigma}^2_{C,\langle q,u\rangle}}$ is the sample variance optimal weighting.*

3.4 Discussion

Operating in the hybrid model is most beneficial utility-wise if the opt-in user records and client user records come from the same distribution – i.e., the two groups have fairly similar observed search behavior. If that is not the case, the differential privacy guarantees still hold, but the accuracy of BLENDER's estimates may decrease.

Improvement in utility over what can be achieved in the local model comes from two sources: the hybrid privacy model lets us develop a better algorithm for client data collection and the analysis of algorithms' variances lets us smartly combine the results.

In practice, a system for local search or trend computation would be run at regular intervals in order to refresh the data as well as accommodate for users being added to, removed from, or moving between the opt-in and the client groups. We have focused on the problem of obtaining local search or trend computation results for a single execution of the system. While one could simply re-run BLENDER at regular intervals to obtain new results (with potentially different opt-in and client groups), this comes at a cost to privacy. We leave the task of improving the temporal aspect of BLENDER beyond what is achievable with standard composition techniques of differential privacy [11] to future work.

4 Experimental Evaluation

We designed BLENDER with an eye toward preserving the utility of the eventual results in the two applications we explore in this paper: trend computation and local search, as described in Section 1.2. We use two established domain-specific utility metrics to assess the utility, the L1 metric and NDCG.

L1: L1 is the Manhattan distance between the estimate and actual probability vectors, in other words, $L1 = \sum_i |\hat{p}_i - p_i|$. The smaller the L1, the better.

NDCG: NDCG is a standard measure of search quality [20, 37] that takes into account the order of queries by performing *discounting*. In particular, most popular queries at the *head* of the search have a higher weight, whereas the relative significance of the less popular queries is reduced. The relevance, or gain, of an item at position i in the ranked list is measured using a graded relevance score defined as $rel_i = \frac{n_i}{\sum_j n_j}$, where n_j is the number of occurrences of the item in position j in the given dataset. The closer i's estimated rank is to

	AOL	Yandex
Data set on disk	1.75 GB	16 GB
Unique queries	4,811,646	13,171,961
Unique clients	519,371	4,970,073
Unique URLs	1,620,064	12,702,350

Figure 8: Data set statistics.

its true rank, the larger the gain. For a *head* of k top elements, the estimated rank list is computed as $DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)}$. Here, the discounting happens because of the $\log_2(i)$ factor that diminishes the effect of later queries. This value is normalized by the Ideal DCG ($IDCG_k$), in which the estimated and the actual ranking are exactly the same, to obtain a value that ranges between 0 and 1.

Since we operate on records rather than just queries, we utilize a generalization of the traditional NDCG score. Here, we compute the NDCG of each query's URL list, $NDCG^q$, as specified above, and then compute the DCG of the queries as $DCG_k^Q = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)} \cdot NDCG^i$.

The final NDCG computation is DCG_k^Q normalized by the analogous Ideal DCG ($IDCG_k^Q$). In a way, our computation considers an NDCG of NDCGs, which makes it even harder for us to maintain consistently high NDCG values when compared to the query-only case. This formulation takes the *ranking* and probabilities from the data set into account, not the actual score that our algorithm outputs.

Since changes to the score may not result in ranking changes, L1 is an even less forgiving measure than NDCG.

Since the purpose of Blender is to estimate probabilities of the top records, we discard the artificially added $\star$ queries and URLs and rescale rel_i prior to L1 and NDCG computations. However, since we use the method of [38] in BlendProbabilities, the probability estimates involving $\star$ have a minor implicit effect on the L1 and NDCG scores.

4.1 Experimental Setup

Data sets: For our experiments, we use the AOL search logs, released in 2006 and the Yandex search data set[5], released in 2013. Figure 8 describes their characteristics.

Data analysis: To familiarize the reader with the approach we used for assessing result quality, Figure 9 shows the top-10 most frequent queries in the AOL data set, with the estimates given by the different "ingredients" of Blender.

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

Query	AOL dataset prob	Blender estimate $\hat{p}_q$	Opt-in estimate $\sum_u \hat{p}_{O,\langle q,u\rangle}$	Client estimate $\hat{p}_{C,q}$	Client estimate $\sum_u \hat{p}_{C,\langle q,u\rangle}$
$\star$	0.9108	0.9103	0.9199	0.9100	0.1468
google	0.0213	0.0216	0.0213	0.0217	0.0216
yahoo	0.0067	0.0070	0.0046	0.0073	**0.0325**
google.com	0.0067	0.0056	0.0023	0.0061	0.0194
myspace.com	0.0057	0.0052	0.0022	0.0057	**0.0258**
mapquest	0.0054	0.0051	**0.0062**	0.0053	0.0192
yahoo.com	0.0043	0.0043	0.0021	0.0048	0.0192
www.google.com	0.0034	0.0004	0.0004	0.0032	0.0098
myspace	0.0033	**0.0034**	**0.0042**	**0.0035**	**0.0255**
ebay	0.0028	0.0026	0.0028	0.0028	0.0254

Figure 9: Comparison of probability estimates for top-10 most popular AOL queries. Parameter choices are shown in Figure 10.

The table is sorted by column 2, which contains the non-private, empirical probabilities from the AOL data set with 1 random record sampled from each user. Column 3 contains the final query probability estimates outputted by Blender. Each algorithm computes probability estimates over the records in the head list; to obtain query probability estimates from these, we simply aggregate the probabilities associated with each URL for a given query (columns 4 and 6). The sample variance of these aggregated probabilities, used for blending, is naively computed as in Theorem 4. Column 5 is the EstimateClientProbabilities' estimate of the query probabilities, since it directly computes these values. Although column 6 is not used for blending in trend computation (where only query probability estimates are produced), columns 4, 5, and 6 are used by the full Blender algorithm when it comes to blending records. Regressions, i.e., estimates that appear out of order relative to column 2, are shown in red.

The biggest takeaway is that the numbers in columns 2 and 3 are similar to each other, with only one regression after Blender's usage. Blender compensates for the weaknesses of both the opt-in and the client estimates. Specifically, despite the opt-in group having several regressions, combining the opt-in and client-data results in only one.

Despite the high number of regressions for the client algorithm's aggregated record probability estimates (Column 6), its query probability estimates (Column 5) only generate one regression. This demonstrates the usefulness of deploying a two-stage reporting process in the client algorithm (first report a query and then the URL), thus allowing for separate estimates of query and record probabilities.

4.2 Experimental Results

We formulate questions for our evaluation as follows: how to choose Blender's parameters (Section 4.2.1), how does Blender perform compared to alternatives (Section 4.2.2), and how robust are our findings (Section 4.2.3)?

4.2.1 Algorithmic and Parameter Choices

BLENDER has a handful of parameters, some of which can be viewed as given externally (by the laws of nature, so to speak), and others whose choice is purely up to the entity that's utilizing BLENDER. We now describe and, whenever possible, motivate, our choices for these.

Privacy parameters, ϵ and δ: Academic literature on differential privacy views the selection of the ϵ parameter as a "social question" [9] and thus uses ϵ in the range of 0.01 to 10 for evaluating algorithm performance (see Table 1 in [18]). The two known industry deployments of differential privacy (by Google [13] and Apple [17]) do not reveal the parameters used, though reverse-engineering of Apple's data by [25] suggests they use $\epsilon = 1$ and $\epsilon = 4$. The work most similar to ours [34] evaluates its algorithm using ϵ in the range $[1, 10]$. We use $\epsilon = 4$ for our experiments, unless otherwise stated. Similarly, there's a range of δ used for evaluating algorithm performance (e.g., $10^{-6}, 10^{-5}, 10^{-4}$ in [26] and 0.05 in [6]). We use a smaller δ for the Yandex dataset than for the AOL dataset to reflect that the Yandex dataset contains data of more users.

We use the same ϵ and δ values for the opt-in and client users. From a behavioral perspective, this reduces a user's opt-in decision down to one purely of trust towards the curator.

Opt-in and client group sizes, $|O|$ and $|C|$: The relative sizes of opt-in group and client group, $|O|$ and $|C|$, respectively, can be viewed as exogenous variables which are dictated by the trust that users place in the search engine[6]. We choose 5% and 2.5% for the fraction of opt-in users as compared to total users as these seem reasonable for representing the fraction of "early adopters" who are willing to supply their data for the improvement of products and allow us to demonstrate the utility benefits of algorithms designed to operate in the hybrid privacy model.

The number of records to collect from each opt-in user, $m_O = 1$: This is mandated by the privacy constraints of CREATEHEADLIST algorithm.

Remaining parameter choices (m_C, f_C, f_O, M) are driven purely by utility considerations.

The number of records to collect from each client, $m_C = 1$: Across a range of experimental values, collecting 1 record per user always yielded greatest utility, justifying this parameter choice. Two factors account for this: 1) the privacy budget must be split across a client's reports, and 2) the accuracy of our algorithm relies on uncorrelated reports, which may not be the case in practice within a given user's set of records.

How to split the privacy budget between query and url reporting for clients, $f_C = 0.85$: Figure 11 shows the effects of the budget split on both the L1 and NDCG metrics. Unsurprisingly, Figure 11a shows that the larger the fraction of client algorithm's budget dedicated to query estimation as opposed to URL estimation, the better the L1 score for the client and BLENDER results. The NDCG metric in Figure 11b shows a trade-off that emerges as we assign more budget to the queries, de-emphasizing the URLs; before and after 0.85, we start seeing a drop in NDCG values for the client algorithm. The orange opt-in line in Figure 11b is constant, as the opt-in group is not affected by the budget split. Somewhat surprisingly with this parameter setting, the NDCG for BLENDER result is also consistently high (nearly equal to and hidden by the opt-in line) and is unaffected by the budget split, unlike the L1 metric.

What fraction of opt-in data to use for creating the headlist, $f_O = 0.95$: Our goal is to build a large candidate head list, and unless we allocate most of the opt-in user data to building such a head list (algorithm CREATEHEADLIST), our subsequent results may be accurate but apply only to a small number of records. Since our opt-in group's size is small relative to our client group size, and it is difficult to generate a head list in the local privacy model – it makes sense to utilize most of the opt-in group's data for the task that is most difficult in the local model. Through experiment we observe that increasing f_O past 95% gives diminishing returns for increasing the head list size; on the other hand, there is a significant utility gain (NDCG and L1) from the use of a small fraction of opt-in users for estimating probabilities of the head list. Thus, rather than using the entire opt-in group for head list generation (i.e., $f_O = 1$), we reserve 5% of the opt-in data for probability estimation.

What should be the final size of the set for which we provide probability estimates, M: The choice of M is influenced by competing considerations. The larger the head list for which we provide the probability estimates, the more effective the local search application (subject to those probability estimates being accurate). However, as desired head list size increases, the accuracy of our estimates drops (most notably due to client data privatization). We want to strike a balance that allows us

[6] If differential privacy gains widespread adoption, it is conceivable that the values of the differential privacy parameters will affect the relative sizes of the groups; for example, the smaller the ϵ, the more users are willing to "opt-in".

Parameter	AOL	Yandex
ϵ	4	4
δ	10^{-5}	10^{-7}
$\frac{\lvert O\rvert}{\lvert O\rvert+\lvert C\rvert}$	5%	2.5%
m_O	1	1
m_C	1	1
f_O	0.95	0.95
f_C	0.85	0.85
M	50	500

Figure 10: Experimental parameters.

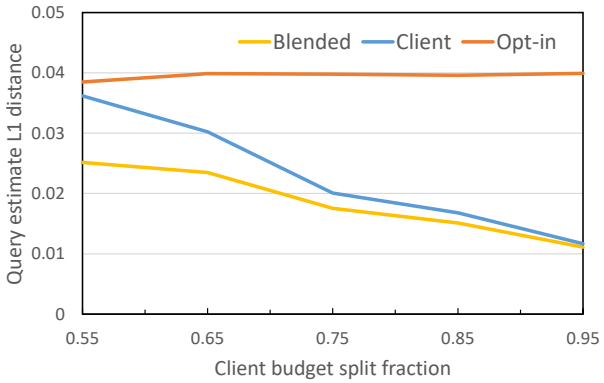
to get a sensibly large record set with reasonably accurate probability estimates it. We choose $M = 50$ and $M = 500$ for the AOL and Yandex datasets, to reflect their differing sizes.

Subsequently, we use the parameters shown in Figure 10 unless explicitly stated.

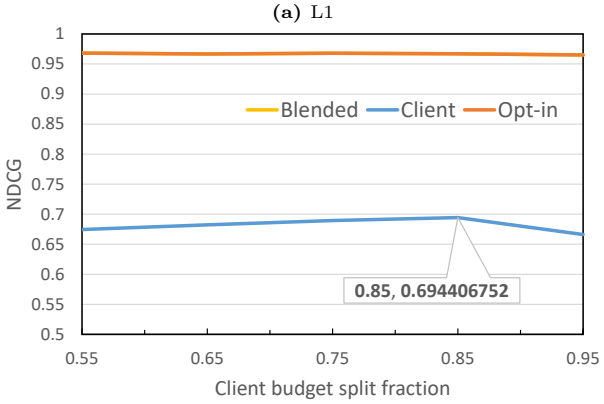
4.2.2 Utility Comparison to Alternatives

The closest related work is a recent paper by Qin *et. al.* [34] in which they provide a utility evaluation of their state-of-the-art algorithm under the local model on the AOL data set for the headlist size of 10. Given the NDCG data that they make available in the paper, we perform a direct comparison with BLENDER across ϵ values. We plot the outcome of the comparison in Figure 12, which shows the NDCG values achieved by BLENDER and by Qin *et. al.* [34] for ϵ values between 1–5. Across the entire range of the privacy parameter, our NDCG values are above 95%, whereas the reported NDCG values for Qin *et. al.* are in the 30% range, at best. We believe that given the intense focus on search optimization in the field of information retrieval, NDCG values as low as those of Qin *et. al.* are generally unusable. Overall, BLENDER significantly outperforms what we believe to be the closest related research project.

Qin *et. al.* and this work use different scoring functions. Qin *et. al.* use a relevance score based purely on the rank of queries in the original AOL data set; this results in penalizing misranked queries regardless of how similar their underlying probabilities may be. BLENDER's relevance scoring only relies on the underlying probabilities, so misranked items with similar underlying probabilities only have a small negative impact on the overall NDCG score; we believe this choice is justified. While it yields increased NDCG scores, BLENDER operates on records (rather than queries, as Qin *et. al.* does). Because of this, the "NDCG of NDCGs" score used to evaluate BLENDER (Section 4) is a strictly less forgiving metric than the traditional NDCG score. Thus, although simultaneously compensating for both differences would yield the ideal comparison, the comparison in Figure 12 is reasonable.



(a) L1



(b) NDCG

Figure 11: Comparing AOL data set results across a range of budget splits for client, opt-in, and blended results.

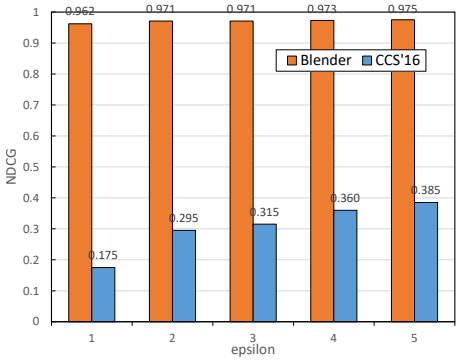


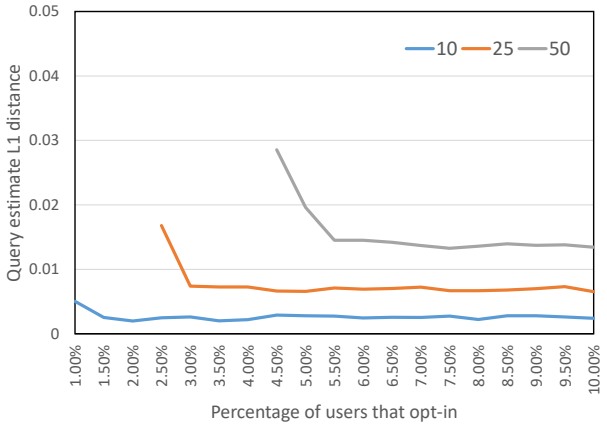
Figure 12: Comparing to the results in the CCS'16 paper by Qin *et. al.* across a range of ϵ values; head list size=10.

4.2.3 Robustness

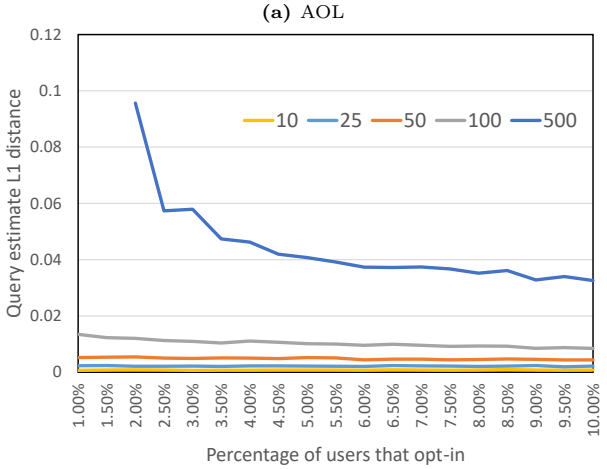
We now discuss how the size of the opt-in group and the choice of ϵ affect BLENDER's utility.

Evaluation of trend computation: Figure 13[7] shows the L1 values as a function of the opt-in percentage ranging between 1% and 10%. We believe

[7]Portions of lines do not appear on figures if the desired head list size was not reached, e.g., in Figure 13a, the line for a head list of size 50 does not begin until 4.5% because that size head list was not created with a smaller opt-in percentage.

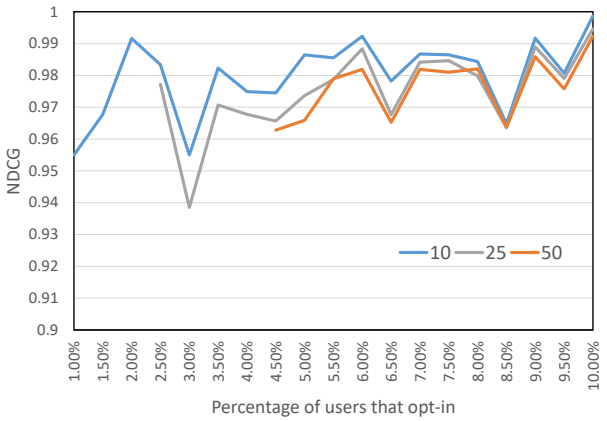


(a) AOL

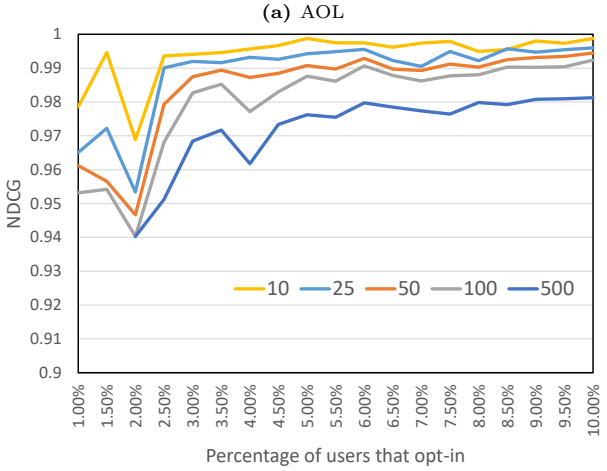


(b) Yandex

Figure 13: L1 as a function of the opt-in percentage.



(a) AOL



(b) Yandex

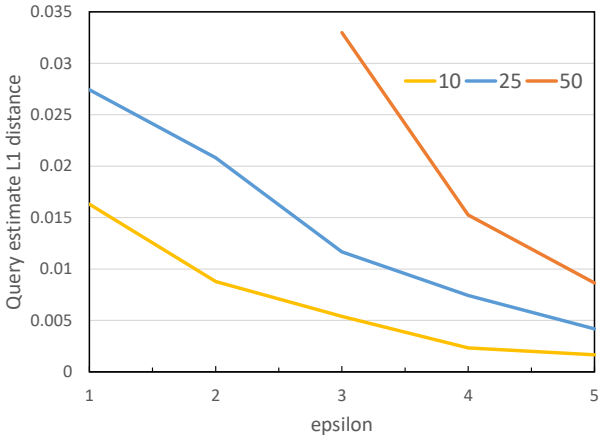
Figure 14: NDCG as a function of the opt-in percentage.

that requiring opt-in percentages in excess of 10% is likely to put undue strain on the system in terms of recruitment; simply put, finding enough opt-in users may prove difficult or impossible in the long run. We see slight differences in the two data sets and across the various head list sizes. Some of the differences might be due to the fact that given the relatively small size of the AOL data set, we need to consider higher opt-in percentages to get reasonably sized head lists and L1 values. In fact, when we increase the opt-in percentage to 10% for the AOL data set, we see a decline in L1 values similar to what is observed in Figure 13b for the Yandex data set. If our goal is to have head lists of 500+, we see that with the larger Yandex data set, an opt-in percentage as small as 2.5% is sufficient.

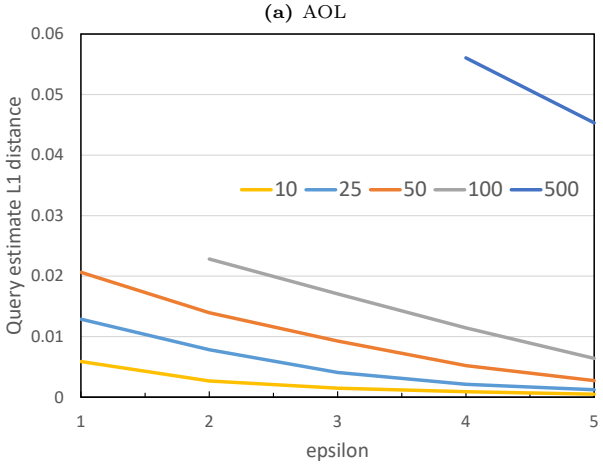
Figure 15 shows the L1 values as a function of ϵ, ranging from 1 to 5. For both data sets, we see a steady decline in the L1 metric, despite aggregating L1 values over longer estimate vectors. With more data in the Yandex data set, we are able to hit small values of L1 (under 0.1) with $\epsilon \geq 1$.

Evaluation of local search computation: Figure 14 shows the NDCG measurements as a function of the opt-in percentage ranging between 1% and 10%. The results are quite encouraging; for the smaller AOL data set, for instance, we need to have an opt-in level of ≈5% to achieve an NDCG level of 95%, which we regard as acceptable. However, for the larger Yandex data set, we hit that NDCG level even sooner: the NDCG value for 1.5% is above 95% for all but the largest head list size.

Figure 16 shows how the NDCG values vary across the two data sets, AOL and Yandex, for a range of head list sizes and ϵ values. We see a clear trend toward higher NDCG values for Yandex, which is not surprising given the sheer volume of data. For the Yandex data set, we can keep ϵ as low as 1 and still achieve NDCG values of 95% and above for all but the two largest head list sizes. For those, we must increase ϵ in order to generate larger head lists from the opt-in users.

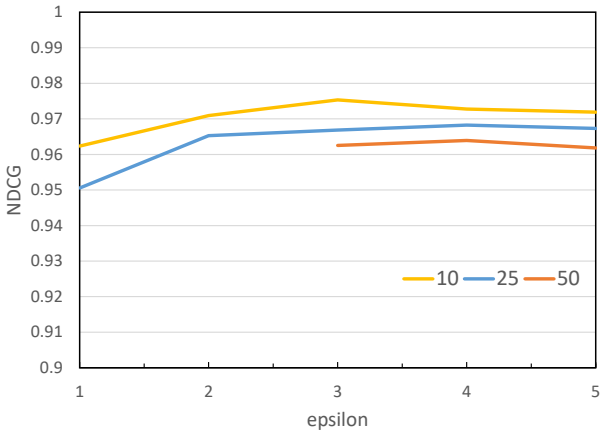


(a) AOL

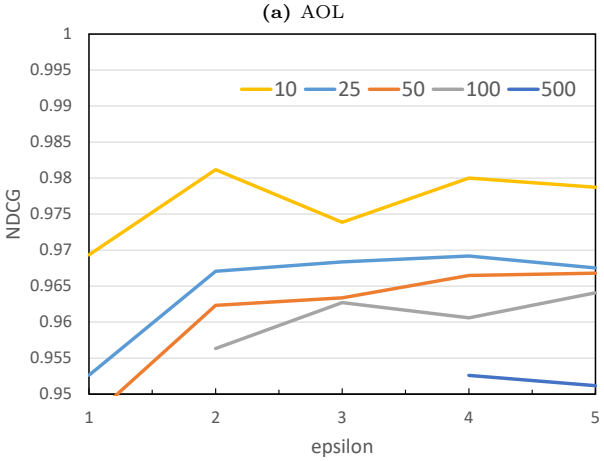


(b) Yandex

Figure 15: L1 statistics for AOL and Yandex data sets for various head list sizes and a range of ϵ values.



(a) AOL



(b) Yandex

Figure 16: NDCG statistics for AOL and Yandex data sets for various head list sizes and a range of ϵ values.

5 Related Work

Algorithms for the trusted curator model: Researchers have developed numerous differentially private algorithms operating in the trusted curator model that result in useful data for a variety of applications. For example, [24, 26, 16, 31] address the problem of publishing a subset of the data contained in a search log with differential privacy guarantees; [27] and [6] propose approaches for frequent item identification; [14] propose an approach for monitoring aggregated web browsing activities; and so on.

Algorithms for the local model: Although the demand for privacy-preserving algorithms operating in the local model has increased in recent years, particularly among practitioners [17, 35], fewer such algorithms are known [39, 19, 8, 13, 5]. Furthermore, the utility of the resulting data obtained through these algorithms is significantly limited compared to what is possible in the trusted curator model, as shown experimentally [15, 21] and theoretically [22].

Our contribution: Our work significantly improves upon the known results by developing application-specific local privatization algorithms that work in combination with the trusted curator model algorithms. Specifically, our insight of providing all users with differential privacy guarantees but achieving it differently depending on whether or not they trust the data curator, enables an efficient privacy-preserving head list construction. The subsequent usage of this head list in the algorithm operating in the local model helps overcome one of the main challenges to utility of privacy-preserving algorithms in the local model [15]. Moreover, the weighted aggregation of probability estimates obtained from algorithms operating in the two models (that explicitly factors in the amount of noise each contributed), enabled remarkable utility gains compared to usage of one algorithm's estimates. As discussed in Section 4.2.2, we significantly outperform the most recently introduced local algorithm of [34]

on metrics of utility in the search context.

6 Conclusions

We proposed a hybrid privacy model and a blended approach that operates within it that combines the upsides of two common models of differential privacy: the local model and the trusted curator model. Using local search as a motivating application, we demonstrated that our proposed approach leads to a significant improvement in terms of utility, bridging the gap between theory and practicality.

Future work: We plan to continue this work in two directions: first, to address any systems and engineering challenges to BLENDER's adoption in practice, including those that arise due to data changing over time; and second, to develop algorithms for other settings where the hybrid privacy model is appropriate, thus facilitating adoption of differential privacy in practice by minimizing the utility impact of privacy-preserving data collection.

References

[1] ACQUISTI, A., BRANDIMARTE, L., AND LOEWENSTEIN, G. Privacy and human behavior in the age of information. *Science 347*, 6221 (2015), 509–514.

[2] ACQUISTI, A., AND GROSSKLAGS, J. Privacy and rationality in individual decision making. *IEEE Security and Privacy (S&P) 2*, 2005 (2005), 24–30.

[3] BAEZA-YATES, R., GIONIS, A., JUNQUEIRA, F., MURDOCK, V., PLACHOURAS, V., AND SILVESTRI, F. The impact of caching on search engines. In *ACM SIGIR Conference on Research and Development in Information Retrieval* (2007), pp. 183–190.

[4] BAEZA-YATES, R., GIONIS, A., JUNQUEIRA, F. P., MURDOCK, V., PLACHOURAS, V., AND SILVESTRI, F. Design trade-offs for search engine caching. *ACM Transactions on the Web 2*, 4 (2008), 20.

[5] BASSILY, R., AND SMITH, A. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Symposium on Theory of Computing (STOC)* (2015), pp. 127–135.

[6] BHASKAR, R., LAXMAN, S., SMITH, A., AND THAKURTA, A. Discovering frequent patterns in sensitive data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)* (2010), pp. 503–512.

[7] DIENLIN, T., AND TREPTE, S. Is the privacy paradox a relic of the past? an in-depth analysis of privacy attitudes and privacy behaviors. *European Journal of Social Psychology 45*, 3 (2015), 285–297.

[8] DUCHI, J. C., JORDAN, M. I., AND WAINWRIGHT, M. J. Local privacy and statistical minimax rates. In *Symposium on Foundations of Computer Science (FOCS)* (2013), pp. 429–438.

[9] DWORK, C. A firm foundation for private data analysis. *Communications of the ACM 54*, 1 (2011), 86–95.

[10] DWORK, C., MCSHERRY, F., NISSIM, K., AND SMITH, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)* (2006), pp. 265–284.

[11] DWORK, C., AND ROTH, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science 9*, 3–4 (2014), 211–407.

[12] DWORK, C., ROTHBLUM, G. N., AND VADHAN, S. Boosting and differential privacy. In *Symposium on Foundations of Computer Science (FOCS)* (2010), pp. 51–60.

[13] ERLINGSSON, Ú., PIHUR, V., AND KOROLOVA, A. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the Conference on Computer and Communications Security (CCS)* (2014), pp. 1054–1067.

[14] FAN, L., BONOMI, L., XIONG, L., AND SUNDERAM, V. Monitoring web browsing behavior with differential privacy. In *Proceedings of the 23rd international conference on World wide web (WWW)* (2014), pp. 177–188.

[15] FANTI, G., PIHUR, V., AND ERLINGSSON, Ú. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies (PETS) 2016*, 3 (2016), 41–61.

[16] GÖTZ, M., MACHANAVAJJHALA, A., WANG, G., XIAO, X., AND GEHRKE, J. Publishing search logs–a comparative study of privacy guarantees. *IEEE Transactions on Knowledge and Data Engineering 24*, 3 (2012), 520.

[17] GREENBERG, A. Apple's differential privacy is about collecting your data – but not your data. In *Wired* (June 13, 2016).

[18] HSU, J., GABOARDI, M., HAEBERLEN, A., KHANNA, S., NARAYAN, A., PIERCE, B. C., AND ROTH, A. Differential privacy: An economic method for choosing epsilon. In *27th IEEE Computer Security Foundations Symposium (CSF)* (2014), pp. 398–410.

[19] HSU, J., KHANNA, S., AND ROTH, A. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming (ICALP)* (2012), pp. 461–472.

[20] JÄRVELIN, K., AND KEKÄLÄINEN, J. Cumulated gain-based evaluation of ir techniques. *Transactions on Information Systems (TOIS) 20*, 4 (2002), 422–446.

[21] KAIROUZ, P., BONAWITZ, K., AND RAMAGE, D. Discrete distribution estimation under local privacy. In *Proceedings of the International Conference on Machine Learning (ICML)* (2016), pp. 2436–2444.

[22] KAIROUZ, P., OH, S., AND VISWANATH, P. Extremal mechanisms for local differential privacy. In *Advances in Neural Information Processing Systems (NIPS)* (2014), pp. 2879–2887.

[23] KOROLOVA, A. Privacy violations using microtargeted ads: A case study. *Journal of Privacy and Confidentiality 3*, 1 (2011), 27–49.

[24] KOROLOVA, A. *Protecting Privacy when Mining and Sharing User Data*. PhD thesis, Stanford University, 2012.

[25] KOROLOVA, A. *Differential Privacy in iOS 10*, Sep 13, 2016. https://twitter.com/korolova/status/775801259504734208.

[26] Korolova, A., Kenthapadi, K., Mishra, N., and Ntoulas, A. Releasing search queries and clicks privately. In *Proceedings of the International Conference on World Wide Web (WWW)* (2009), pp. 171–180.

[27] Li, N., Qardaji, W., Su, D., and Cao, J. Privbasis: frequent itemset mining with differential privacy. *Proceedings of the VLDB Endowment 5*, 11 (2012), 1340–1351.

[28] Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J., and Vilhuber, L. Privacy: Theory meets practice on the map. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE)* (2008), pp. 277–286.

[29] Madden, M., and Rainie, L. Americans' attitudes about privacy, security and surveillance. Tech. rep., Pew Research Center, 2015.

[30] McSherry, F., and Talwar, K. Mechanism design via differential privacy. In *Symposium on Foundations of Computer Science (FOCS)* (2007), pp. 94–103.

[31] Meng, X., Xu, Z., Chen, B., and Zhang, Y. Privacy-preserving query log sharing based on prior n-word aggregation. In *Trustcom/BigDataSE/IâĂŃ SPA, 2016 IEEE* (2016), IEEE, pp. 722–729.

[32] Merriman, C. Microsoft reminds privacy-concerned Windows 10 beta testers that they're volunteers. In *The Inquirer, http://www.theinquirer.net/2374302* (Oct 7, 2014).

[33] Narayanan, A., and Shmatikov, V. Robust deanonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy* (2008), pp. 111–125.

[34] Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., and Ren, K. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the Conference on Computer and Communications Security (CCS)* (2016), pp. 192–203.

[35] Shankland, S. How google tricks itself to protect chrome user privacy. In *CNET* (Oct 31, 2014).

[36] Silvestri, F. Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval 4*, 1–2 (2010), 1–174.

[37] Valizadegan, H., Jin, R., Zhang, R., and Mao, J. Learning to rank by optimizing NDCG measure. In *Advances in neural information processing systems (NIPS)* (2009), pp. 1883–1891.

[38] Wang, W., and Carreira-Perpinán, M. A. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541* (2013).

[39] Warner, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association 60*, 309 (1965), 63–69.

[40] White House Report. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *Journal of Privacy and Confidentiality 4*, 2 (2012), 95–142.

Appendices

A Client Data Algorithm

A.1 Privacy

Theorem 3. LocalAlg *is* (ϵ, δ)*-differentially private.*

Proof. We show this by proving that each iteration of the `for` loop in line 7 of LocalAlg is (ϵ', δ')-differentially private, where $\epsilon' = \epsilon/m_C$ and $\delta' = \delta/m_C$. Since there are at most m_C iterations of this loop for each client, composition of differentially private mechanisms [12] guarantees that LocalAlg ensures (ϵ, δ)-differential privacy for each client.

Denote each iteration of the `for` loop in line 7 of LocalAlg by L; it takes as input a record $\langle q, u\rangle \in D$, and returns a record, which we denote $L(\langle q, u\rangle)$. If q is not in HL or u is not in $HL[q]$, then they immediately get transformed into a default value $(\star)$ that is in the head list. Since L outputs only values that exist in the head list, to confirm differential privacy we need to prove that for any arbitrary neighboring data sets $\langle q, u\rangle$ and $\langle q', u'\rangle$, $\Pr\big[L(\langle q, u\rangle) \in Y\big] \leq e^{\epsilon'} \Pr\big[L(\langle q', u'\rangle) \in Y\big] + \delta'$ holds for all sets of head list records Y.

Whenever $k = 1$ or $k_q = 1$, the only query (or URL for a specific query) is $\star$, which will be output with probability 1. Thus, differential privacy trivially holds, since the reported values then do not rely on the client's data. Thus, we'll assume $k \geq 2$ and $k_q \geq 2$. Note that there is a single decision point where it is determined whether q will be reported truthfully or not. Thus, we can split the privacy analysis into two parts: 1) Usage of the f_C fraction of the privacy budget to report a query, and 2) Usage of the remainder of the privacy budget to report a URL (given the reported query). This decomposes a simultaneous two-item (ϵ', δ') reporting problem into two single-item reporting problems with (ϵ'_Q, δ'_Q) and (ϵ'_U, δ'_U) respectively, where $\epsilon'_Q = f\epsilon'$, $\delta'_Q = f\delta'$, $\epsilon'_U = (1 - f_C)\epsilon'$, and $\delta'_U = (1 - f_C)\delta'$.

1. Privacy of Query Reporting:

Consider the query-reporting case first. Overloading our use of L, let $L(q)$ be the portion of L that makes use of q. We first ensure that

$$\Pr[L(q) = q_{HL}] \leq \exp(\epsilon'_Q) \Pr[L(q') = q_{HL}] + \frac{\delta'_Q}{2} \qquad (1)$$

holds for all q, q', and $q_{HL} \in HL$. This trivially holds when $q_{HL} = q = q'$ or $q_{HL} \notin \{q, q'\}$. The remaining scenarios to consider are: 1) $q \neq q_{HL}, q' = q_{HL}$ and 2) $q = q_{HL}, q' \neq q_{HL}$. By the design of the algorithm, $\Pr[L(q_{HL}) = q_{HL}] = t$ and $\Pr[L(\bar{q}_{HL}) =$

$q_{HL}] = (1-t)(\frac{1}{k-1})$, where $\bar{q}_{HL}$ represents any query not equal to q_{HL}. With $t = \frac{\exp(\epsilon'_Q)+(\delta'_Q/2)(k-1)}{\exp(\epsilon'_Q)+k-1}$, it is simple to verify that inequality (1) holds.

Consider an arbitrary set of head list queries Y.

$$\begin{aligned}
&\Pr[L(q) \in Y] = \sum_{q_{HL} \in Y} \Pr[L(q) = q_{HL}] \\
&= \sum_{q_{HL} \in Y \setminus \{q,q'\}} \Pr[L(q) = q_{HL}] + \sum_{q_{HL} \in Y \cap \{q,q'\}} Pr[L(q) = q_{HL}] \\
&= \sum_{q_{HL} \in Y \setminus \{q,q'\}} \Pr[L(q') = q_{HL}] + \sum_{q_{HL} \in Y \cap q,q'} \Pr[L(q) = q_{HL}] \qquad (2) \\
&\leq \sum_{q_{HL} \in Y \setminus \{q,q'\}} \Pr[L(q') = q_{HL}] + \sum_{q_{HL} \in Y \cap \{q,q'\}} \left(e^{\epsilon'_Q} \Pr[L(q') = q_{HL}] + \frac{\delta'_Q}{2}\right) \qquad (3) \\
&\leq e^{\epsilon'_Q} \sum_{q_{HL} \in Y} \Pr[L(q') = q_{HL}] + 2 \cdot \frac{\delta'_Q}{2} \\
&= e^{\epsilon'_Q} \Pr[L(q') \in Y] + \delta'_Q,
\end{aligned}$$

Equality (2) stems from the fact that the probability of reporting a false query is independent of the user's true query. The inequality (3) is a direct application of inequality (1). Thus, L is (ϵ'_Q, δ'_Q)-differentially private for query-reporting.

2. Privacy of URL Reporting:

With t_q defined as $t_q = \frac{\exp(\epsilon'_U)+0.5\delta'_U(k_q-1)}{\exp(\epsilon'_U)+k_q-1}$, an analogous argument shows that the (ϵ'_U, δ'_U)-differential privacy constraints hold if the original q is kept. On the other hand, if it is replaced with a random query, then they trivially hold as the algorithm reports a random element in the URL list of the reported query, without taking into consideration the client's true URL u.

By composition [12], each of the at most m_C iterations of L is $(\epsilon'_Q+\epsilon'_U, \delta'_Q+\delta'_U) = (\epsilon', \delta')$-differentially private. □

A.2 Denoising

Observation 1. *$\hat{p}_C$ gives the unbiased estimate of record and query probabilities under* EstimateClientProbabilities.

Proof. Reporting records is a two-stage process (first, decide which query to report, then report a record); similarly, denoising is also done in two stages.

Denoising of query probability estimates: Let $r_{C,q}$ denote the probability that the algorithm has received query q as a report, and let p_q be the true probability of a user having query q. We want to learn p_q based on $r_{C,q}$. By the design of our algorithm, $r_{C,q} = t \cdot p_q + \sum_{q' \neq q} p_{q'}(1-t)\frac{1}{k-1} = t \cdot p_q + \frac{1-t}{k-1}\sum_{q' \neq q} p_{q'} = t \cdot p_q + \frac{1-t}{k-1}(1-p_q)$.

Solving for p_q in terms of $r_{C,q}$ yields $p_q = \frac{r_{C,q}-\frac{1-t}{k-1}}{t-\frac{1-t}{k-1}}$. Using the obtained data for the query $\hat{r}_{C,q}$, we estimate $p_{C,q}$ as $\hat{p}_{C,q} = \frac{\hat{r}_{C,q}-\frac{1-t}{k-1}}{t-\frac{1-t}{k-1}}$.

Denoising of record probability estimates: Analogously, denote by $r_{C,\langle q,u\rangle}$ the probability that the algorithm has received a record $\langle q, u\rangle$ as a report, and recall $p_{\langle q,u\rangle}$ is the record's true probability in the data set. Then $r_{C,\langle q,u\rangle} = t \cdot t_q \cdot p_{\langle q,u\rangle} + \left(t\frac{1-t_q}{k_q-1}\right)(p_q - p_{\langle q,u\rangle}) + \left(\frac{1-t}{k-1}\frac{1}{k_q}\right)(1-p_q)$, recalling from the algorithm that k_q is the number of URLs associated with query q and t_q is the probability of truthfully reporting u given that query q was reported. Solving for $p_{\langle q,u\rangle}$ yields $p_{\langle q,u\rangle} = \frac{r_{C,\langle q,u\rangle}-\left(t\frac{1-t_q}{k_q-1}p_q+\frac{(1-t)(1-p_q)}{(k-1)k_q}\right)}{t(t_q-\frac{1-t_q}{k_q-1})}$.

Using the obtained data for the empirical report estimate $\hat{r}_{C,\langle q,u\rangle}$ together with the query estimate $\hat{p}_{C,q}$, we estimate $p_{\langle q,u\rangle}$ as $\hat{p}_{C,\langle q,u\rangle} = \frac{\hat{r}_{C,\langle q,u\rangle}-\left(t\frac{1-t_q}{k_q-1}\hat{p}_{C,q}+\frac{(1-t)(1-\hat{p}_{C,q})}{(k-1)k_q}\right)}{t(t_q-\frac{1-t_q}{k_q-1})}$. □

B Blending

Theorem 4. *When $m_O = 1$ the unbiased variance estimate for* EstimateOptinProbabilities *can be computed as:* $\hat{\sigma}^2_{O,\langle q,u\rangle} = \frac{|D_T|}{|D_T|-1}\left(\frac{\hat{p}_{O,\langle q,u\rangle}(1-\hat{p}_{O,\langle q,u\rangle})}{|D_T|} + 2\left(\frac{b_T}{|D_T|}\right)^2\right)$.

Proof. Given the head list, the distribution of EstimateOptinProbabilities' estimate for a record $\langle q, u\rangle$ is given by $r_{O,\langle q,u\rangle} = p_{\langle q,u\rangle} + \frac{Y}{|D_T|}$, where $Y \sim$ Laplace(b_T) where b_T is the scale parameter and $|D_T|$ is the total number of records from the opt-in users used to estimate probabilities. The empirical estimator for $r_{O,\langle q,u\rangle}$ is $\hat{r}_{O,\langle q,u\rangle} = \frac{1}{|D_T|}\sum_{j=1}^{|D_T|} X_j + Y$, where $X_j \sim$ Bernoulli$(p_{\langle q,u\rangle})$ is the random variable indicating whether report j was record $\langle q, u\rangle$.

The expectation of this estimator is given by $E[\hat{r}_{O,\langle q,u\rangle}] = p_{\langle q,u\rangle}$. Thus, $\hat{r}_{O,\langle q,u\rangle}$ is an unbiased estimator for $p_{\langle q,u\rangle}$. We denote $\hat{p}_{O,\langle q,u\rangle} = \hat{r}_{O,\langle q,u\rangle}$ to explicitly reference it as the estimator of $p_{\langle q,u\rangle}$. The variance for this estimator is

$$\begin{aligned}
\sigma^2_{O,\langle q,u\rangle} &= V[\hat{p}_{O,\langle q,u\rangle}] = V\Big[\frac{1}{|D_T|}\Big(\sum_{j=1}^{|D_T|} X_j + Y\Big)\Big] \\
&= \frac{1}{|D_T|^2}\Big(V\big[\sum_{j=1}^{|D_T|} X_j\big] + V[Y]\Big) \qquad (4) \\
&= \frac{1}{|D_T|^2}\Big(\sum_{j=1}^{|D_T|} V[X_j] + V[Y]\Big) \qquad (5) \\
&= \frac{1}{|D_T|^2}\big(|D_T| \cdot p_{\langle q,u\rangle}(1-p_{\langle q,u\rangle})\big) + 2\Big(\frac{b_T}{|D_T|}\Big)^2 \\
&= \frac{p_{\langle q,u\rangle}(1-p_{\langle q,u\rangle})}{|D_T|} + 2\Big(\frac{b_T}{|D_T|}\Big)^2.
\end{aligned}$$

Equality 4 comes from the independence between Y and all X_j. Equality 5 relies on an assumption of independence between X_j, X_k for all $j \neq k$ (i.e., the iid assumption discussed prior to the theorem statements in Section 3.3).

To actually compute this variance, we need to use the data in place of the unknown $p_{\langle q,u\rangle}$. Using $\hat{p}_{O,\langle q,u\rangle}$ directly in place of $p_{\langle q,u\rangle}$ requires a $\frac{|D_T|}{|D_T|-1}$ factor correction (known as "Bessel's correction[8]") to generate an unbiased estimate. Thus, the variance of each opt-in record probability estimate is: $\hat{\sigma}^2_{O,\langle q,u\rangle} = \frac{|D_T|}{|D_T|-1}\left(\frac{\hat{p}_{O,\langle q,u\rangle}(1-\hat{p}_{O,\langle q,u\rangle})}{|D_T|} + 2\left(\frac{b_T}{|D_T|}\right)^2\right)$. □

Note that in line 15 of EstimateOptinProbabilities, the use of this sample variance expression in re-computing $\hat{\sigma}^2_{O,\langle\star,\star\rangle}$ is not statistically valid, so our computation of $\hat{p}_{O,\langle\star,\star\rangle}$ and $\hat{p}_{\langle\star,\star\rangle}$ is sub-optimal. Despite that, our overall utility, which does not include $\star$, is good (see Section 4).

Theorem 5. *When $m_C = 1$ the unbiased variance estimate for* EstimateClientProbabilities *can be computed as:* $\hat{\sigma}^2_{C,\langle q,u\rangle} = \frac{1}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2}\cdot\Big(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|-1} + \big(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\big)^2\hat{\sigma}^2_{C,q} + \frac{2|D_C|}{|D_C|-1}\big(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\big)\big(\frac{k-2+t}{kt-1}\big)\hat{r}_{C,\langle q,u\rangle}\Big)$.

Proof. From Section A.2 on denoising, the distribution of the reported query q from the client mechanism is given by $r_{C,q} = t\cdot p_q + \frac{1-t}{k-1}(1-p_q)$, and so the true probability of query q is distributed as $p_q = \frac{r_{C,q} - \frac{1-t}{k-1}}{t - \frac{1-t}{k-1}}$. The empirical estimator for p_q is $\hat{p}_{C,q} = \frac{\hat{r}_{C,q} - \frac{1-t}{k-1}}{t-\frac{1-t}{k-1}}$, where $\hat{r}_{C,q}$ is the empirical estimator of $r_{C,q}$ defined explicitly as $\hat{r}_{C,q} = \frac{1}{|D_C|}\sum_{j=1}^{|D_C|} X_j$, where $X_j \sim \text{Bernoulli}(r_{C,q})$ is the random variable indicating whether report j was query q and $|D_C|$ is the total number of records from the client users.

The variance of $\hat{r}_{C,q}$ is

$$\begin{aligned} V[\hat{r}_{C,q}] &= V\Big[\frac{1}{|D_C|}\sum_{j=1}^{|D_C|} X_j\Big] \\ &= \Big(\frac{1}{|D_C|}\Big)^2 \sum_{j=1}^{|D_C|} V[X_j] \qquad (6) \\ &= \big(\frac{1}{|D_C|}\big)^2\big(|D_C|\cdot r_{C,q}(1-r_{C,q})\big) = \frac{r_{C,q}(1-r_{C,q})}{|D_C|}, \end{aligned}$$

where equality 6 relies on an assumption of independence between X_j, X_k for all $j \neq k$ (i.e., the iid assumption discussed prior to the theorem statements in Section 3.3).

[8] https://en.wikipedia.org/wiki/Bessel's_correction

Then, the variance of $\hat{p}_{C,q}$ is

$$\sigma^2_{C,q} = V[\hat{p}_{C,q}] = V\Big[\frac{\hat{r}_{C,q} - \frac{1-t}{k-1}}{t-\frac{1-t}{k-1}}\Big] = \frac{r_{C,q}(1-r_{C,q})}{|D_C|\big(t-\frac{1-t}{k-1}\big)^2}.$$

To actually compute this variance, we need to use the data in place of the unknown $r_{C,q}$. Using $\hat{r}_{C,q}$ directly in place of $r_{C,q}$ requires including Bessel's $\frac{|D_C|}{|D_C|-1}$ factor correction to yield an unbiased estimate. Thus, the variance of the query probability estimates by the client algorithm is: $\hat{\sigma}^2_{C,q} = \left(\frac{1}{t-\frac{1-t}{k-1}}\right)^2 \frac{\hat{r}_{C,q}(1-\hat{r}_{C,q})}{|D_C|-1}$.

Using a similar procedure for records we obtain the unbiased variance estimate as $\hat{\sigma}^2_{C,\langle q,u\rangle} = \frac{1}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2}\cdot\Big(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|-1} + \big(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\big)^2\hat{\sigma}^2_{C,q} + \frac{2|D_C|}{|D_C|-1}\big(\frac{1-t}{(k-1)k_q} - \frac{t-tt_q}{k_q-1}\big)\big(\frac{k-2+t}{kt-1}\big)\hat{r}_{C,\langle q,u\rangle}\Big)$. □

Theorem 6. *If $\hat{\sigma}^2_{O,\langle q,u\rangle}$ and $\hat{\sigma}^2_{C,\langle q,u\rangle}$ are sample variances of $\hat{p}_{O,\langle q,u\rangle}$ and $\hat{p}_{C,\langle q,u\rangle}$ respectively, then $w_{\langle q,u\rangle} = \frac{\hat{\sigma}^2_{C,\langle q,u\rangle}}{\hat{\sigma}^2_{O,\langle q,u\rangle}+\hat{\sigma}^2_{C,\langle q,u\rangle}}$ is the sample variance optimal weighting.*

Proof. With the variance estimates for each algorithm fully computed, a blended estimate of $p_{\langle q,u\rangle}$ is given by $\hat{p}_{\langle q,u\rangle} = w_{\langle q,u\rangle}\cdot\hat{p}_{O,\langle q,u\rangle} + (1-w_{\langle q,u\rangle})\cdot\hat{p}_{C,\langle q,u\rangle}$, which has sample variance $\hat{\sigma}^2_{\langle q,u\rangle} = w^2_{\langle q,u\rangle}\cdot\hat{\sigma}^2_{O,\langle q,u\rangle} + (1-w_{\langle q,u\rangle})^2\cdot\hat{\sigma}^2_{C,\langle q,u\rangle}$. Minimizing $\hat{\sigma}^2_{\langle q,u\rangle}$ with respect to $w_{\langle q,u\rangle}$ yields the desired. □