Journal of Privacy and Confidentiality
Vol. 9 (2) 2019 TPDP 2017
Submitted Aug 2, 2018
Published Sept 2019

BLENDER: ENABLING LOCAL SEARCH WITH A HYBRID DIFFERENTIAL PRIVACY MODEL

BRENDAN AVENT*, ALEKSANDRA KOROLOVA*, DAVID ZEBER, TORGEIR HOVDEN, AND BENJAMIN LIVSHITS

University of Southern California
e-mail address: bavent@usc.edu

University of Southern California
e-mail address: korolova@usc.edu

Mozilla
e-mail address: dzeber@mozilla.com

Mozilla
e-mail address: torgeir@eritreum.com

Imperial College London
e-mail address: 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 that improves the utility of obtained data, while providing users with their desired privacy guarantees. We apply this algorithm to the task of privately computing the head of the search log and show that the blended approach provides significant improvements in the utility of the data compared to related work. Specifically, on two large search click datasets, 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 organizations are confronted with the tension between collection and sharing of mass-scale user data to fuel innovations and user's privacy. Today, an organization that needs user data to improve the quality of its service often has no choice but to perform its own data collection. However, its users may not want to share their raw data with the organization, especially if they consider it to be sensitive. Furthermore, by collecting this user data, the organization assumes liability, as it may be leaked through security breaches,

Key words and phrases: differential privacy, local search, search log.

The preliminary version of this work appeared at the *26th USENIX Security Symposium* in 2017 [6].

*Supported in part by NSF grant #1755992 and a gift from Mozilla.

www.journalprivacyconfidentiality.org
DOI:10.29012/jpc.680

required to be shared through 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 of the data sharing and use, but also from minimizing the raw data collected by the organization to achieve their goal. Thus, data collection with differential privacy in the local model is the best match for user expectations of privacy and the guarantees an organization may want to provide. However, due to the amounts of data required in order to achieve meaningful utility when ensuring privacy in the local model, such collection is relevant only to the biggest organizations with massive user bases, and the smaller ones get edged out. The goal of this work is to open possibilities for use of differential privacy to include organizations with smaller user bases.

Local differential privacy: Over the last several years, we have seen some examples of the local differential privacy (LDP) model beginning to be used for data collection in practice, most notably in the context of the Chrome web browser [18] and Apple's data collection [22].

In the LDP model, the data collector (such as Google or Apple) obtains aggregate data statistics without observing the exact values of user's private data. This is achieved by applying a privacy-preserving perturbation to each user's raw data before it leaves the user's device. This approach protects not only the individual users, but also the data collector from risks such as data breaches.

Trusted curator model: An alternative model that has been most commonly used in the academic literature on differential privacy to date is the *trusted curator model*, where a curator first collects each user's private data and then produces and releases a privacy-preserving version of the collected dataset. In this model, although users are guaranteed that the released dataset 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.

Hybrid model: The contribution of this paper stems from our observation that the two models can co-exist. People's attitudes toward privacy vary widely [3, 2, 12], and some users may be comfortable with sharing their data with a trusted curator, while others may require the privacy protections of the local model.

In industry practice, many companies already 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]"; Microsoft states that "[Windows Insider Program] services may automatically collect and provide data to Microsoft, which may include your personal information[2]"; Google has a similar provision for the beta testers of the Canary build of the Chrome browser[3].

For these users (referred to as the *opt-in group*), the trusted curator privacy model is a natural match. For all other users (referred to as *clients*), the local privacy model is appropriate. Our goal is to demonstrate that by separating the user pool into these two

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

[2] https://insider.windows.com/en-us/program-agreement/

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

groups, according to their trust (or lack thereof) in the data aggregator, we can improve the utility of the collected data while preserving privacy. We dub this new model the *hybrid differential privacy* model.

Applications: We consider two specific applications in this paper to demonstrate the usefulness of the hybrid model: *local search* provided by a browser and *search trend computation*.

Local search revolves around the problem of how a browser maker can collect information about users' clicks as they interact with search engines[4] in order to create the *head* of the search logs, i.e., the collection of the most popular queries and their corresponding URLs, to be made available to users *locally* (i.e., on their devices). 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. With proper privacy measures in place, the head of the search logs can then 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 also 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 local caching of the most frequently posed search queries has a disproportionately positive impact on the expected query latency [35, 7], as search engine queries follow a power-law distribution [8].

Search trend computation entails finding the most popular queries and sorting them in order of popularity. An example of this is the Google trends service[5], which has an up-to-date list of trending topics and queries.

Utility challenges: Local search and search trend computation can be thought of as problems in the category of *heavy hitter discovery and estimation*, which is a well-studied problem in the context of information retrieval. Heavy hitter discovery is also one of the canonical problems in privacy-preserving data analysis [11, 30]. Moreover, the recent work in the LDP model is focused on precisely that problem [18, 34, 19] or very closely related ones of histogram computations [10, 25]. However, current privacy-preserving approaches in the local model lead to utility losses that are quite significant, to a point where the results are no longer useful for local search. For instance, it is common to seek NDCG [23, 37] values above 0.9 for the local search problem of finding the most popular queries; however, the current best algorithm applied to this problem under the LDP model [34] is only able to attain an NDCG value of 0.385 while ensuring LDP with an ϵ of 5 (see Section 4.3.2 for further detail).

If privacy constraints make the utility too low compared to the original, the privacy-preserving approach is at risk to not be adopted. This is especially true in the context of search tasks, where users have been conditioned for years to expect high-quality results.

1.1. **Contributions.** Our work makes the following contributions:

- Introduces and utilizes a realistic, hybrid trust model, which removes the traditional "all-or-nothing" trust assumption towards a central curator.

[4]A browser maker may choose to combine the results obtained from user interactions that stem from several search engines depending on the context or surface results obtained from Baidu and not Bing depending on the user's current location.

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

- Proposes BLENDER, an algorithm that takes advantage of the hybrid differential privacy model for computing heavy hitters. Specifically, BLENDER utilizes data obtained from the opt-in users in order to modify the privacy-preserving algorithm run for all other users and then combines the data of opt-in and all other users in an informed way, in order to improve the utility of the privacy-preserving computation.
- Performs a comprehensive utility evaluation of BLENDER on two large search click datasets, comprising 1.75 and 16 GB for two applications: search trend computation and local search. Demonstrates that BLENDER achieves high levels of utility (i.e., NDCG values in excess of 95%) while maintaining differential privacy for reasonable privacy parameter values.
- Provides the first empirical demonstration that hybrid trust models, such as those combining data provided in the local model of differential privacy with data provided in the trusted curator model, can lead to non-trivial improvements in utility. Thus, it suggests the exploration of algorithms for such models as a promising direction for increasing the feasibility of differential privacy's deployment by both a wider range of organizations as well as for a wider variety of applications.

2. Overview

We now discuss the curator models that will form the basis of our hybrid model in more detail, provide a high-level overview of our proposed algorithm, BLENDER, that coordinates the privatization, collection and aggregation of data in this model, and discuss some of the specific choices we make in this algorithm. We use the application of enabling local search based on user search histories while preserving differential privacy throughout; but, as will become clear from the discussion, our approach can be applied to other frequency-based discovery and estimation tasks.

2.1. 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. The problem stems from the impossibility of foreseeing all attacks of adversaries capable of utilizing outside knowledge. Differential privacy, which has become the gold standard privacy guarantee in the academic literature, and is gaining traction in industry and government [18, 22, 31], 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.

Most differentially private algorithms developed to date [16] operate in the *trusted curator model*: all users' private data is collected by the curator before privatization techniques are applied to it. This means that although the privacy of the eventual result of the computation is ensured, the curator gets to observe the users' private data. However, as was most recently argued by Apple [22], users may not trust the data collector with their data, and may prefer privatization to occur before their data reaches the collector. This is known as the *local model*, since privatization occurs locally.

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 biased 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* [40], guarantees each person plausible deniability and is differentially private (with privacy parameters determined by the bias of the coin). But since the randomness is incorporated into the algorithm in a precisely specified way, the data collector is able to recover an estimate of the true proportion of HIV-positive people if enough people contribute their locally privatized data.

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.

Formally, an algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private [15] 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 to differences in 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 one of the user's values may be altered. Concretely, for the case of collecting a single search record from each user, the databases in the trusted curator model contain a collection of search records and differ in the value of one record, while the databases in the local model contain one record each.

2.2. **An Algorithm for the Hybrid Model.** 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. Our proposed algorithm, Blender, coordinates the privatization, collection, and aggregation of the data from the opt-in and the client users.

2.2.1. *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 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 algorithm 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.

The core of our approach is in utilizing the strengths of each of the models. Specifically, the *head list discovery* portion of the task – that is, finding the names of the most-frequent queries and URLs – can be done much more effectively under the trusted curator model than under the local model. Thus, we assign most opt-in users to this task. With the domain significantly narrowed, the remaining users are then assigned to the *frequency estimation* portion of the task, where the underlying frequencies of the queries and URLs are estimated.

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, then (optionally) trims the candidate head list down further creating the final head list. This result of this component of the algorithm 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, as well as with the world.

Each member of the client group receives the head list obtained from the opt-in group. Each client then individually uses the head list to apply a differential privacy-preserving perturbation to their data, subsequently reporting their perturbed results to Blender.

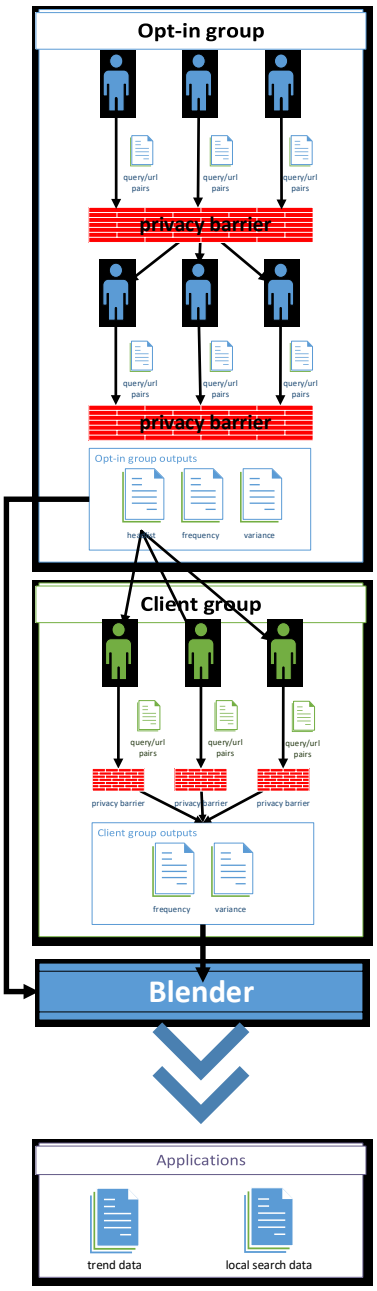


Figure 1. Architectural diagram of Blender's processing steps.

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.

Finally, 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 using their respective variance estimates. BLENDER outputs the obtained records and their combined record estimates, which can then be used to drive local search, determine trends, and more.

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, the post-processing property of differential privacy [16] guarantees that as long as CREATEHEADLIST, ESTIMATEOPTIN-PROBABILITIES, 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.

2.2.2. *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 algorithm follows the strategy introduced in [29] by aggregating the records of the opt-in users from S, and including those records whose noisy count exceeds a threshold in the head list. The noise to add to the true counts[6] and the threshold are calibrated to ensure differential privacy, using [28]. 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.

[6] $\text{Lap}(b)$ refers to a random draw from the Laplace distribution with scale b.

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

Our algorithm differs from previous work in two 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 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 [28].

For those records included in the initial head list set, the algorithm described in Figure 4 uses the remaining opt-in users' data (from set T) to differentially privately estimate each record's probability, denoted $\hat{p}_O$. The M most frequent records in $\hat{p}_O$ are retained to form the final head list. This algorithm is the standard Laplace mechanism from the differential privacy literature [15], with scale of noise calibrated 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.

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.

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 chosen by BLENDER's developer to optimize utility goals, as will be discussed in Section 4.3.1.

CreateHeadList$(\epsilon, \delta, S, m_O)$

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

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

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

Algorithms for client data collection (Figures 5, 6): Figure 5 defines the algorithm for the client group. Here, records are no longer treated as a single entity, but rather in a two-stage process: first privatizing the query, then privatizing the URL. This helps optimize utility in the setting where the number of queries is significantly larger than the number of URLs associated with each query. Privatization as achieved by following a generalization of the randomized response mechanism introduced by [40], and utilizes the head list obtained from the opt-in group in order to perform the privatization locally by each client. At its core, the privatization is achieved by reporting the true record with a certain bounded probability, and otherwise, randomizing the report among all the other records in the head list.

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. This allows use of 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 [19, 34]. Another distinction from the standard randomized response mechanism is our utilization of δ.

Note that the choices of m_C and f_C are not related to privacy constraints, and can be chosen by Blender's developer to optimize utility goals, as will be discussed in Section 4.3.1.

The local nature of the reporting, using a randomization procedure that can report any record with some probability, induces a predictable bias to the distribution of reported records. To account for this, a denoising procedure must be performed in order to compute proper estimates.

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 $\text{Lap}(b_T)$.
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.

These probability estimates, denoted $\hat{p}_C$, along with variance estimates are then passed to the BlendProbabilities part of Blender. The technical discussion of the algorithm's privacy properties and variance estimate computations follow in Sections 3.2 and 3.3.

Algorithm for Blending (Figure 7): The blending portion of the algorithm 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 algorithm respectively added. This produces a blended probability estimates $\hat{p}$ 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 this algorithm follows in Section 3.3.

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

1: Append query $q = \star$ to HL.
2: **for** each query $q \in HL$ **do**
3: Append URL $u = \star$ to $HL[q]$.
4: **for** each client $i \in C$ **do**
5: **let** $D_{C,i} = \text{LocalAlg}(\epsilon, \delta, m_C, f_C, HL)$ be the reports from i's local execution of LocalAlg.
6: **let** D_C be the concatenation of all reported client datasets, $D_{C,i}$.
7: Denote $|D_C|$ as the total number of records in D_C.
8: **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.
9: **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).
10: **for** $q \in HL$ **do**
11: **let** $\hat{r}_{C,q}$ be the fraction of queries q in D_C.
12: $\hat{p}_{C,q} = \frac{\hat{r}_{C,q} - \frac{1-t}{k-1}}{t - \frac{1-t}{k-1}}$
13: $\hat{\sigma}^2_{C,q} = \frac{1}{\left(t - \frac{1-t}{k-1}\right)^2} \frac{\hat{r}_{C,q}(1-\hat{r}_{C,q})}{|D_C|-1}$
14: **for** $u \in HL[q]$ **do**
15: **let** $\hat{r}_{C,\langle q,u\rangle}$ be the fraction of records which are $\langle q, u\rangle$ in D_C.
16:

$$p_{C,\langle q,u\rangle} = \frac{\hat{r}_{C,\langle q,u\rangle} - \left(t\frac{1-t_q}{k_q-1}\hat{p}_{C,q} + \frac{1-t}{k-1}\frac{1}{k_q}(1-\hat{p}_{C,q})\right)}{t(t_q - \frac{1-t_q}{k_q-1})}$$

17:

$$\hat{\sigma}^2_{C,\langle q,u\rangle} = \frac{|D_C|}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2(|D_C|-1)} \cdot$$

$$\Big(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|} + \big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)^2\hat{\sigma}^2_{C,q} + 2\big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)\big(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,q})}{|D_C|(t-\frac{1-t}{k-1})}\big)\Big)$$

18: **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.

3. Technical Details

We now present further technical details related to the instantiations of the sub-algorithms for Blender, such as privacy proofs and the motivation for BlendProbabilities.

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

let $D_{C,i}$ be the database aggregating at most m_C records from current client i.
$\epsilon' = \epsilon / m_C$, and $\delta' = \delta / m_C$.
$\epsilon'_Q = f_C \epsilon', \epsilon'_U = \epsilon' - \epsilon'_Q$ and $\delta'_Q = f_C \delta', \delta'_U = \delta' - \delta'_Q$.
$k = |HL|$, and $t = \frac{\exp(\epsilon'_Q) + (\delta'_Q/2)(k-1)}{\exp(\epsilon'_Q) + k - 1}$.
for each $q \in HL$ **do**:
    $k_q = |HL[q]|$, and $t_q = \frac{\exp(\epsilon'_U) + (\delta'_U/2)(k_q - 1)}{\exp(\epsilon'_U) + k_q - 1}$.
for each $\langle q, u \rangle \in D_{C,i}$ **do**
    if $q \notin HL$ **then**
        Set $q = \star$.
    if $u \notin HL[q]$ **then**
        Set $u = \star$.
    With probability $(1 - t)$,
        let q' be a unif. random query from $HL \setminus q$.
        let u' be a unif. random URL from $HL[q']$.
        report $\langle q', u' \rangle$.
        continue
    With probability $(1 - t_q)$,
        let u' be a unif. random URL from $HL[q] \setminus u$.
        report $\langle q, u' \rangle$.
        continue
    report $\langle q, u \rangle$.

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

3.1. **Opt-in Data Algorithms.** Differential privacy of the algorithms handling opt-in client data follows directly from previous work.

Theorem 3.1 . *([28])* CreateHeadList *guarantees* (ϵ, δ)*-differential privacy if* $m_O = 1, \epsilon > \ln(2)$, *and* $\tau \geq 1$.

Theorem 3.2 . *([15])* 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 prove the privacy and explain the logic behind the aggregation procedure next.

Theorem 3.3 . LocalAlg *is* (ϵ, δ)*-differentially private.*

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 [39]).
6: **return** $\hat{p}$.

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

Proof. See Appendix. □

The reports aggregated by the client algorithm form an empirical distribution over the records (and queries). This distribution is biased in an explicit way, as described by the reporting process, creating a significantly flatter distribution relative to the true underlying distribution. Since the noise addition process is known, the bias is also known, and can be used to "unflatten" the distribution as a post-processing step to obtain a more useful, unbiased estimate of the record distribution. We refer to this as "denoising" the reported empirical distribution $\hat{r}_C$ to obtain the final estimate from the client algorithm, $\hat{p}_C$.

Observation 3.4 . *$\hat{p}_C$ gives the unbiased estimate of record and query probabilities under* ESTIMATECLIENTPROBABILITIES.

Proof. See Appendix. □

3.3. **Blending.** The opt-in algorithm and 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 estimates come from different sources. The error in the estimates obtained from the opt-in algorithm is due to the addition of Laplace noise, whereas the error in the estimates obtained from the client algorithm is due to randomizing the true record over the set of records in the head list. Thus, our goal is to determine the variances of the estimates obtained from the two algorithms and 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 algorithms 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 weight 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 [39]) ensures that the estimates sum to 1 and are non-negative.

Theorem 3.5 presents our computation of the sample variance of EstimateOptinProbabilities, Theorem 3.6 presents our computation of the sample variance of EstimateClientProbabilities, and Theorem 3.7 motivates the weighting scheme used in BlendProbabilities.

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 3.5 . *If $m_O = 1$ then the unbiased variance estimate for the opt-in group's record probabilities 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. See Appendix. □

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 3.6 . *If $m_C = 1$ then the unbiased variance estimate for the client group's record probabilities can be computed as:*

$$\hat{\sigma}^2_{C,\langle q,u\rangle} = \frac{|D_C|}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2(|D_C|-1)}\cdot$$
$$\Big(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|} + \Big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\Big)^2\hat{\sigma}^2_{C,q} + 2\Big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\Big)\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,q})}{|D_C|(t-\frac{1-t}{k-1})}\Big).$$

Proof. See Appendix. □

Theorem 3.7 . *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, and the blended estimate is the convex combination $\hat{p}_{\langle q,u\rangle} = w_{\langle q,u\rangle}\cdot\hat{p}_{O,\langle q,u\rangle} + (1-w_{\langle q,u\rangle})\cdot\hat{p}_{C,\langle q,u\rangle}$, then the sample variance optimal weighting is given by $w_{\langle q,u\rangle} = \frac{\hat{\sigma}^2_{C,\langle q,u\rangle}}{\hat{\sigma}^2_{O,\langle q,u\rangle}+\hat{\sigma}^2_{C,\langle q,u\rangle}}$.*

Proof. See Appendix. □

3.4. **Discussion.** We have intentionally used (slight modifications) of existing algorithms for BLENDER's sub-algorithms in order to demonstrate the power of the blended approach within the hybrid privacy model. However, it is conceivable (see Section 5) that the sub-algorithms themselves can be improved, yielding further improvements in the utility achieved by BLENDER.

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.

4. Experimental Evaluation

4.1. **Utility Metrics.** One pitfall in much of the research in the area of differential privacy is an insufficient emphasis on the utility loss due to privacy constraints. 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. We use two domain-specific utility metrics to assess the loss of 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 [23, 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 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 then 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 frequencies from the dataset 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 [39] in BlendProbabilities, the probability estimates involving $\star$ have a minor implicit effect on the L1 and NDCG scores.

4.2. **Experimental Setup.** For our experiments, we use the AOL search logs, released in 2006 and the Yandex search dataset[7], released in 2013. Figure 8 describes their characteristics.

	AOL	**Yandex**
Dataset 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. Dataset statistics.

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 dataset, with the estimates given by the different "ingredients" of Blender.

The table is sorted by column 2, which contains the non-private, empirical probabilities from the AOL dataset 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 naïvely computed as in Theorem 3.5. Column 5 is the EstimateClientProbabilities' estimate of the query probabilities, since it directly computes these values. While 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 entire 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 Blender's usage resulting in only one regression.

Blender compensates for the weaknesses of both the opt-in and the client estimates. Specifically, despite the opt-in group having several regressions in this particular instance, combining the opt-in and client-data compensates for that, resulting in only one regression.

4.3. **Experimental Results.** We formulate questions for our evaluation as follows: how are Blender's parameters chosen (Section 4.3.1), how does Blender perform compared to alternatives (Section 4.3.2), and how robust are our findings (Section 4.3.3)?

[7] 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** $\sum_u \hat{p}_{C,\langle q,u\rangle}$
$\star$	0.9121	0.9144	0.9148	0.9143	0.9143
google	0.0211	0.0211	0.0220	0.0210	0.0210
yahoo	0.0067	0.0081	0.0061	0.0088	0.0088
google.com	0.0066	0.0075	**0.0083**	0.0073	0.0073
myspace.com	0.0055	0.0046	0.0034	0.0052	0.0052
mapquest	0.0055	**0.0062**	**0.0051**	**0.0066**	**0.0066**
yahoo.com	0.0048	0.0047	**0.0057**	0.0043	0.0043
www.google.com	0.0044	0.0038	0.0043	0.0035	0.0035
myspace	0.0034	0.0030	0.0031	0.0030	0.0030
ebay	0.0030	0.0030	0.0030	0.0029	0.0029

Figure 9. Comparison of probability estimates for top-10 most popular AOL queries. Parameter choices are shown in Figure 11 (except with $\epsilon = 3$ used here).

4.3.1. *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 deploying BLENDER. We now describe and, whenever possible, motivate, our choices for these.

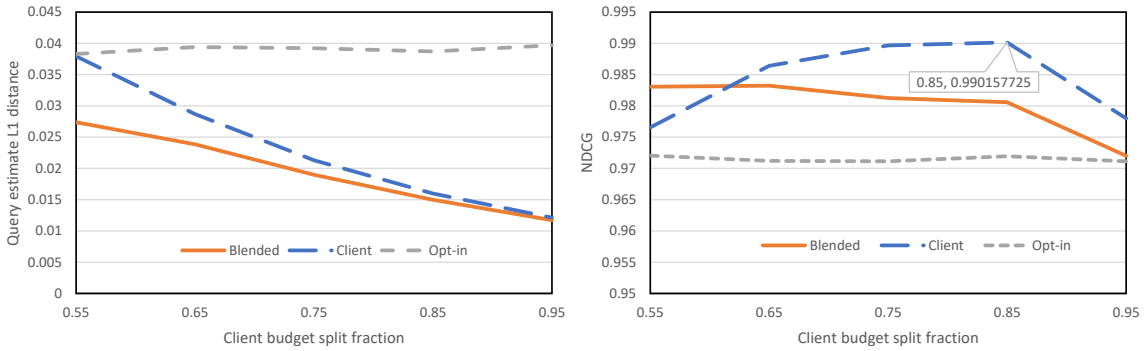
Privacy parameters, ϵ and δ: We view ϵ and δ as privacy parameters given to us externally (e.g., by what is a common practice for differentially private algorithms in the industry [36, 5, 18]). We use a δ that is larger for the AOL dataset than for the Yandex dataset to reflect that the Yandex dataset 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[8]. We choose 5% for AOL and 3% for Yandex for the fraction of opt-in users because they seem reasonable while still allowing us to effectively demonstrate the utility benefits of BLENDER.

The number of records to collect from each opt-in user, $m_O = 1$: This is a choice necessitated by the privacy constraints of the CREATEHEADLIST algorithm.The choices for remaining parameters: 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

[8]In the future, as differential privacy gains widespread adoption, it is conceivable that the values of the privacy parameters may affect their relative sizes; for example, the smaller the ϵ, the more users are willing to "opt-in". However, this relationship is out of the scope of this work.



(a) L1 **(b)** NDCG

Figure 10. Comparing AOL dataset results across a range of budget splits for client, opt-in, and blended results.

reports, and 2) the accuracy of our algorithm relies on uncorrelated reports, an assumption which likely does not hold 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 10 shows the effects of the budget split on both the L1 and NDCG metrics. Unsurprisingly, Figure 10a 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 10b shows a trade-off that emerges as we assign more budget to the queries, de-emphasizing the URLs. The client algorithm NDCG value peaks at a budget split of 0.85; choosing a split above this point induces a significant drop in the blended NDCG values. Note that the grey opt-in line in Figure 10b is constant, as the opt-in group is not affected by the budget split.

What fraction of opt-in data to use for creating the headlist, $f_O = 0.95$: We choose $f_O = 0.95$ because 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, whereas in order to speak of an effective local search application in practice we need to amass a headlist of at least double or triple digits in size. Even without looking at experimental data, this choice makes sense: our opt-in group size is small relative to our client group size, and it is difficult to generate a head list in the local privacy model – so it makes sense to utilize most of the opt-in group's data for the task that is most difficult in the local model.

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 to get a sensibly large record set with reasonably accurate probability estimates for it. We choose $M = 50$ and $M = 500$ for the AOL and

Parameter	AOL	Yandex
ϵ	4	4
δ	10^{-5}	10^{-7}
$\frac{\|O\|}{\|O\|+\|C\|}$	5%	3%
m_O	1	1
m_C	1	1
f_O	0.95	0.95
f_C	0.85	0.85
M	50	500

Figure 11. Default parameters used in experiments.

Yandex datasets, to reflect their differing sizes. Subsequently, we use the parameters shown in Figure 11, unless explicitly stated.

4.3.2. *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 on the AOL dataset 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 dataset; this results in penalizing

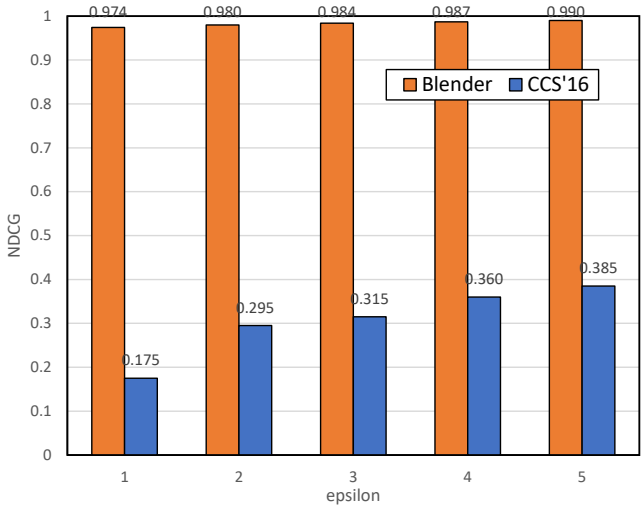
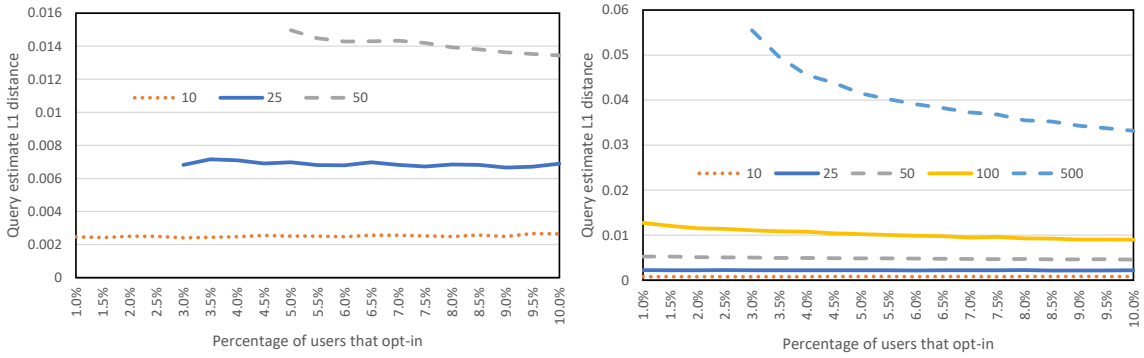
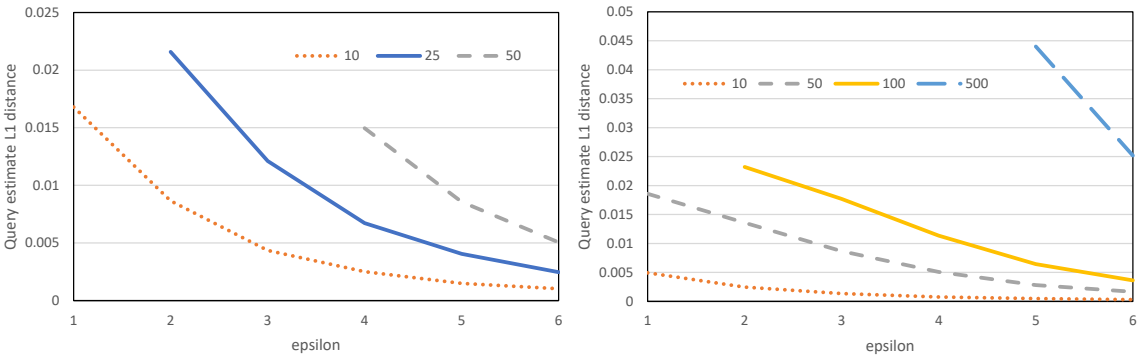


Figure 12. Comparing to the results in the CCS'16 paper by Qin *et. al.* across a range of ϵ values; head list size=10.



(a) AOL **(b)** Yandex

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



(a) AOL **(b)** Yandex

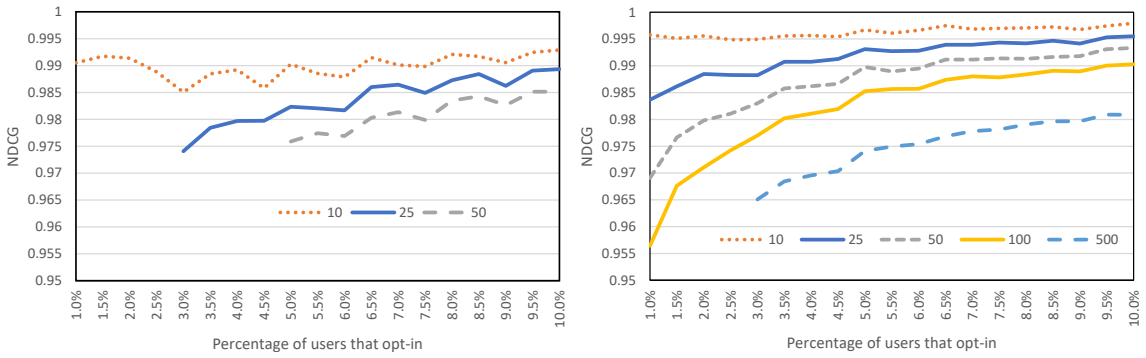
Figure 14. L1 statistics for AOL and Yandex datasets for various head list sizes and a range of ϵ values.

misranked queries regardless of their underlying probabilities. 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 4.1) 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.

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

Evaluation of trend computation: Figure 13[9] shows the L1 values as a function of the opt-in percentage ranging between 1% and 10%. We believe that requiring opt-in percentages

[9]Portions of lines do not appear on figures if the desired head list size was not reached (e.g., in Figure 13a, the line representing results for a head list of size 50 does not begin until 5% because a head list of size 50 could not be generated with a lower opt-in percentage).



(a) AOL **(b)** Yandex

Figure 15. NDCG as a function of the opt-in percentage.

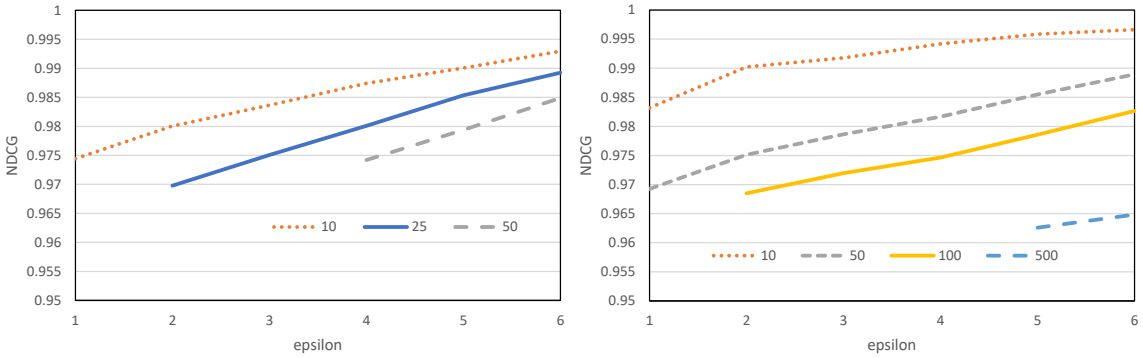
in excess of 10% is likely to put undue strain on the system in terms of recruitment; simply put, finding enough opt-in users may provide difficult or impossible in the long run. We see slight differences in the two datasets and across the various head list sizes. Some of the differences might be due to the fact that given the relatively small size of the AOL dataset, 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 dataset, we see a slight decline in L1 values for the largest head list size similar to what is observed in Figure 13b for the Yandex dataset. If our goal is to have head lists of 500+, we see that with the larger Yandex dataset, an opt-in percentage as small as 3% is sufficient. The main take-away from this is that when the opt-in group is large enough to attain the desired head list size, the trend computation results generally will be high quality in terms of the L1 values.

Figure 14 shows the L1 values as a function of ϵ, ranging from 1 to 6. For both datasets, we see a steady decline in the L1 metric, despite aggregating L1 values over longer estimate vectors. With more data in the Yandex dataset, we are able to hit small values of L1 (under 0.1) with $\epsilon \geq 1$.

Evaluation of local search computation: Figure 15 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 dataset, we need to have an opt-in level of ≈5% to achieve an NDCG level in excess of 95%, which we regard as acceptable. However, for the larger Yandex dataset, we hit that NDCG level even sooner: for an opt-in group composing 1% of the users, the NDCG level is above 95% for all but the largest head list size.

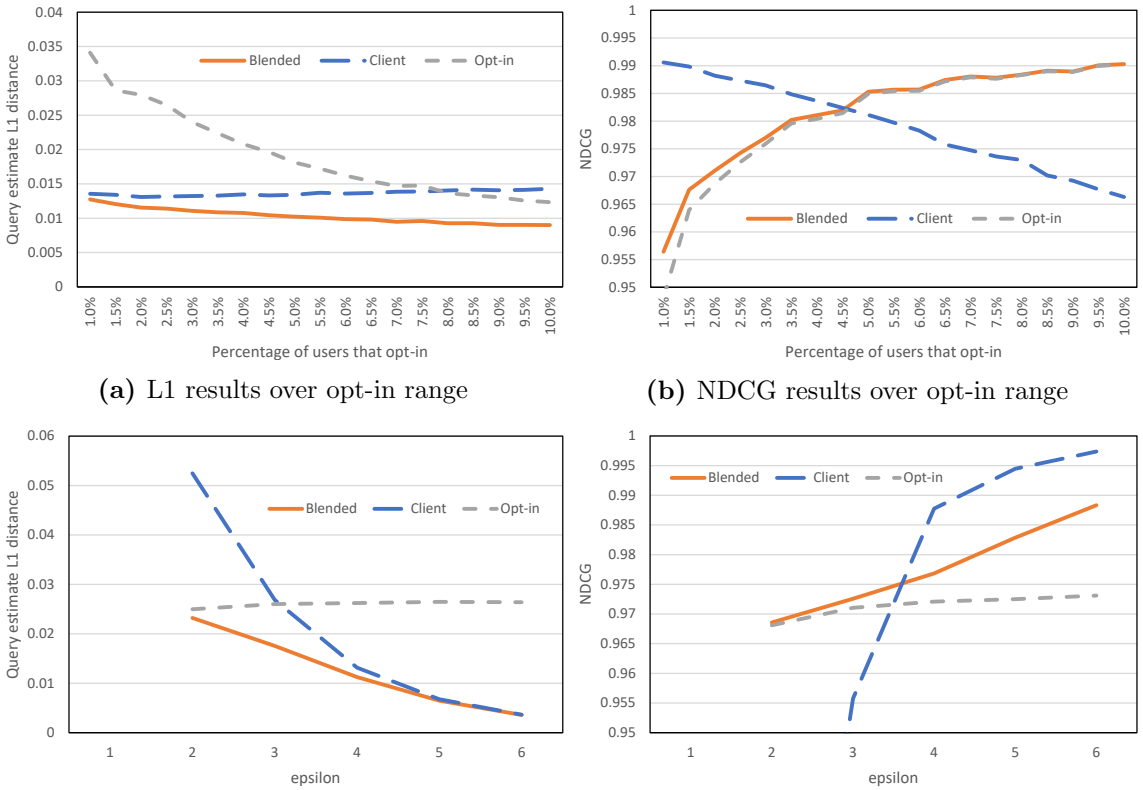
Figure 16 shows how the NDCG values vary across the two datasets, AOL and Yandex, for a range of head list sizes and ϵ values. We see a clear trend toward higher NDCG values for Yandex, which is not surprising given the sheer volume of data. For the Yandex dataset, we can keep ϵ as low as 1 and still 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.

Each group's effect on the blended result: While these blended results demonstrate the algorithms's high-utility capability, one central question remains: to what extent are each



(a) AOL **(b)** Yandex

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



(a) L1 results over opt-in range **(b)** NDCG results over opt-in range

(c) L1 results over ϵ range **(d)** NDCG results over ϵ range

Figure 17. L1 and NDCG statistics broken out between the different groups' results on the Yandex dataset with head list size 100 across a range of opt-in percentages ((a) and (b), with $\epsilon = 4$) and a range of ϵ values ((c) and (d), with 3% opt-in).

group's estimates contributing to the final blended result? Specifically, does the small number of samples with low noise from the opt-in group dominate the large number of samples with high noise from the client group, or vice-versa?

For a head list size 100 on the Yandex dataset, Figure 17 examines this question for a range of opt-in percentages and ϵ values. These graphs show a complex relationship between the two groups' utility with regards to the final blended result. In all cases, the blended result is better than the worse of either the opt-in or client results. With regards to L1 distance, the blended result is better than *both* groups' individual results when varying either the opt-in user percentage or the ϵ value.

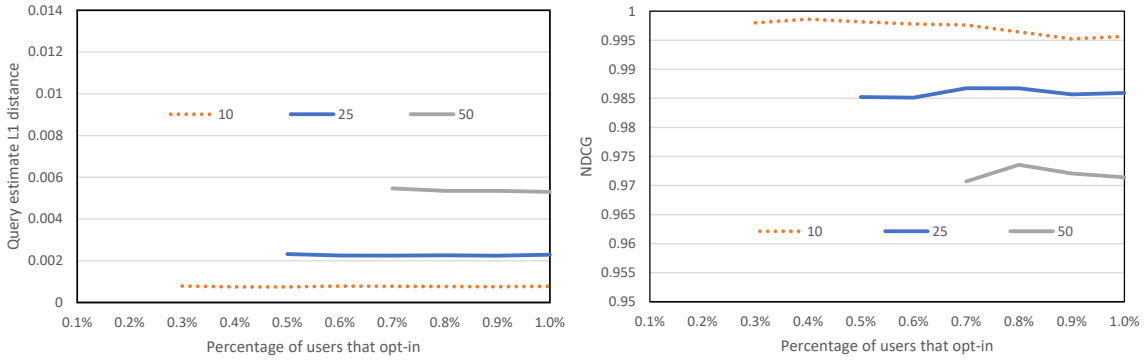
When increasing the opt-in user percentage, the two group's results behave as expected: the opt-in group's results improve as it gains more users, and the client group's results gradually deteriorate as it loses users. Interestingly, Figures 17a and b show that the L1 distance of the client group's query results deteriorate quite slowly as their group size decreases, whereas their NDCG results deteriorate more quickly. To understand this behavior, observe that there are significantly fewer queries (what the query estimate L1 distance is measuring) than there are query-URL pairs (what the NDCG is measuring). Also note that the utility of the randomized response component of the local algorithm degrades as the set of items under consideration increases. These two facts in combination explain the difference in the deterioration rates of the client group's utility between Figure 17a and b.

For the blended result, the NDCG values mainly track the opt-in group's NDCG values even in the case where the client result is clearly better (from 1% up to 3%); this would support the idea that the opt-in results may be dominating the client results when it comes to the blending process. However, this trend doesn't appear to hold when increasing the ϵ, as the blended results rapidly improve with the client results, while the opt-in results remain relatively flat. Interestingly, as ϵ is increased, the opt-in group's L1 results remain relatively constant and its NDCG results only slightly improve. This is caused by the large amount of noise that is inherent in the opt-in group due to its relatively small size; i.e., a 3% sized opt-in group induces a certain level of sampling error such that the noise introduced for privacy is negligible by comparison.

The takeaway is that there is no single clear-cut group that dominates in its contribution to the final blended result; in fact, both groups appear to contribute across the ranges of parameters considered.

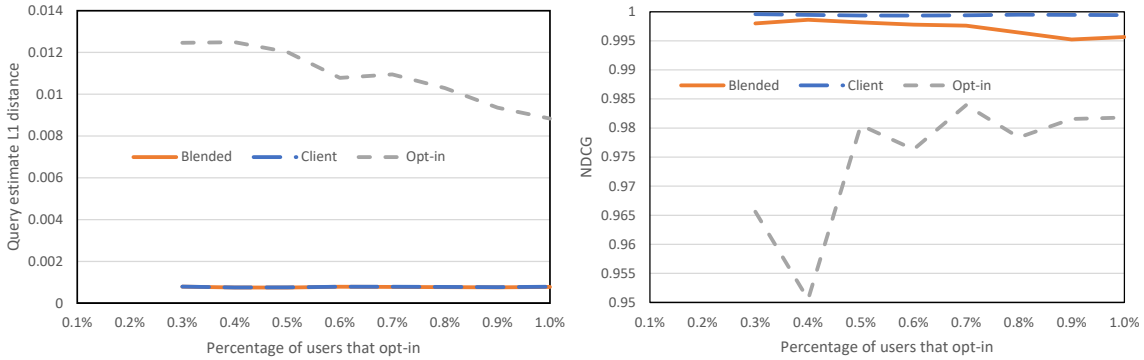
When the opt-in group is tiny: In the real-world, it may be the case that a 5% or even a 3% sized opt-in group is still too large to be considered feasible. As mentioned in the evaluation of trend computation, the utility results are generally good *conditioned on* the desired head list size being achieved. When the opt-in group becomes too small, it becomes a significantly greater challenge to achieve these large head list sizes. For the head list sizes that we can achieve at smaller opt-in percentages, what are the utility results we can expect? Figure 18 shows the performance on the Yandex dataset targeting smaller head list sizes across opt-in group sizes ranging from 0.1% up to 1%. These results confirm our previous conclusion that once a head list size can be attained, getting high utility probability estimates for the records is a significantly easier challenge.

At these tiny opt-in percentages, with 95% of the opt-in group being assigned to head list creation, only 0.005% to 0.05% of the users are estimating the probabilities under the trusted curator model. We must ask: in this setting, to what extent *are* the users contributing to the high-utility blended results? Figure 19 shows the L1 and NDCG values for the opt-in group, client group, and final blended results across these tiny opt-in sizes for a head list of size 10 on the Yandex dataset. As suspected, the estimates from the



(a) L1 results **(b)** NDCG results

Figure 18. L1 and NDCG statistics for the Yandex dataset for various head list sizes across a range of tiny opt-in percentages.



(a) L1 results **(b)** NDCG results

Figure 19. L1 and NDCG statistics broken out per group for the Yandex dataset for head list size 10 and a range of tiny opt-in percentages.

opt-in group have much lower utility relative to the client group. The blending procedure is able to automatically take advantage of the high variance results of the opt-in group (stemming from the tiny number of samples used by this group in estimating the probabilities) and weigh the blending much more heavily towards the client group's estimates.

5. Related Work

Algorithms for the trusted curator model: Researchers have developed numerous privacy-preserving algorithms operating in the trusted curator model that result in useful data for a variety of applications. Specifically, the works of [29, 21, 28] pioneered the study of search log data release with differential privacy guarantees; the works of [30] and [11] proposed approaches for privacy-preserving frequent item identification, and so on.

Algorithms for the local model: Although some progress has been made in developing privacy-preserving algorithms operating in the local model [40, 14, 18, 10], the utility of

the resulting data is limited [19, 25]. Furthermore, it is known that for fixed desired differential privacy parameters, the elimination of the trusted data collector comes at the cost of diminished utility [26, 27]. Very recently, much attention has been given to the heavy hitters problem in the local differential privacy setting both from a theoretical and an applied perspective [38, 9, 24]. Since BLENDER came prior to these recent works, we did not compare our results against theirs[10].

Our contribution: Our work significantly improves upon the known results by developing application-specific local model algorithms that work in combination with 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 local algorithms [19]. As discussed in Section 4.3.2, we significantly outperform previous work of [34] on metrics of utility in the search context.

6. CONCLUSIONS

We proposed a hybrid differential privacy model, which permits a mixture of trust models. In this work, we considered a mix of two primary models studied by the differential privacy community, which differ only in their trust towards the curator: the local model and the trusted curator model. Using local search as a motivating application, we developed and tested an algorithm which demonstrates that operating in the hybrid model enables significant improvements in terms of utility compared to previously known approaches. Thus, we showed that developing algorithms for hybrid models holds promise for decreasing the gap between theory and practicality of differential privacy.

Future work: The primary direction for future work is to better understand the power of the hybrid model. Specifically, what application areas and algorithms can most effectively utilize data submitted in a mixture of trust models, what utility improvements can such algorithms bring, and how do they depend on the underlying data or user group sizes. Recent work by [13] has begun to study this question for the problem of mean estimation. Works by [33] and [20] show that there is much to be gained by combining trusted curator data with public data, giving another example of a hybrid model that holds promise. A related sub-question is understanding the role that interaction between users contributing data in the local model and in the trusted curator model can play in improving utility.

Another important direction for future work is to address the assumption in current work that user data comes from the same distribution regardless of their trust model, which may not hold in practice. As a start, one can differentially privately evaluate whether the distributions are different using a small sample of records from both groups using the techniques of [1, 4]. When and how should the differences between groups be taken into account is an open question.

[10]We note that the new algorithms proposed could be directly applied as the local group's sub-algorithm to improve the utility of BLENDER.

Finally, optimizing the sub-algorithms used by Blender (see Section 3.4) and providing *a priori* estimates of its utility under specific assumptions is a promising direction for advancing the deployment of differentially private algorithms in the hybrid model.

References

[1] J. Acharya, Z. Sun, and H. Zhang. Differentially private testing of identity and closeness of discrete distributions. In *Advances in Neural Information Processing Systems*, pages 6878–6891, 2018.

[2] A. Acquisti, L. Brandimarte, and G. Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

[3] A. Acquisti and J. Grossklags. Privacy and rationality in individual decision making. *IEEE Security and Privacy (S&P)*, 2(2005):24–30, 2005.

[4] M. Aliakbarpour, I. Diakonikolas, and R. Rubinfeld. Differentially private identity and closeness testing of discrete distributions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[5] Apple. Learning with privacy at scale. volume 1. Apple Machine Learning Journal, 2017. https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html.

[6] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits. BLENDER: Enabling local search with a hybrid differential privacy model. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 747–764, Vancouver, BC, 2017. USENIX Association.

[7] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. The impact of caching on search engines. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 183–190, 2007.

[8] R. Baeza-Yates, A. Gionis, F. P. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. Design trade-offs for search engine caching. *ACM Transactions on the Web*, 2(4):20, 2008.

[9] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, 2017.

[10] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 127–135, 2015.

[11] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 503–512, 2010.

[12] T. Dienlin and S. Trepte. 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):285–297, 2015.

[13] Y. Dubey and A. Korolova. The power of the hybrid model for mean estimation. *Privacy in Machine Learning Workshop @NeurIPS, arXiv preprint arXiv:1811.12040*, 2018. https://arxiv.org/abs/1811.12040.

[14] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *Symposium on Foundations of Computer Science (FOCS)*, pages 429–438, 2013.

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

[16] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[17] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *Symposium on Foundations of Computer Science (FOCS)*, pages 51–60, 2010.

[18] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 1054–1067, 2014.

[19] G. Fanti, V. Pihur, and Ú. Erlingsson. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2016(3):41–61, 2016.

[20] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

[21] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs–a comparative study of privacy guarantees. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):520, 2012.

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

[23] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[24] J. Jia and N. Z. Gong. Calibrate: Frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge. *arXiv preprint arXiv:1812.02055*, 2018.

[25] P. Kairouz, K. Bonawitz, and D. Ramage. Discrete distribution estimation under local privacy. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2436–2444, 2016.

[26] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2879–2887, 2014.

[27] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 531–540, 2008.

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

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

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

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

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

[33] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[34] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 192–203, 2016.
[35] F. Silvestri. Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval*, 4(1–2):1–174, 2010.
[36] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang. Privacy loss in Apple's implementation of differential privacy on MacOS 10.12. *arXiv preprint arXiv:1709.02753*, 2017. https://arxiv.org/abs/1709.02753.
[37] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing NDCG measure. In *Advances in Neural Information Processing Systems*, pages 1883–1891, 2009.
[38] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security)*, pages 729–745. USENIX Association, 2017.
[39] W. Wang and M. A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.
[40] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

7. Appendix

Theorem 3.3 . 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 algorithms [17] 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 datasets $\langle q, u\rangle$ 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

$$\Pr[L(q) = q_{HL}] \leq \exp(\epsilon'_Q) \Pr[L(q') = q_{HL}] + \frac{\delta'_Q}{2} \tag{7.1}$$

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 (7.1) holds.

Consider an arbitrary set of head list queries Y.

$$\begin{aligned}
\Pr[L(q) \in Y] &= \sum_{q_{HL} \in Y} \Pr[L(q) = q_{HL}] \\
&= \sum_{q_{HL} \in Y \setminus \{q,q'\}} \Pr[L(q) = q_{HL}] + \sum_{q_{HL} \in Y \cap \{q,q'\}} Pr[L(q) = q_{HL}] \\
&= \sum_{q_{HL} \in Y \setminus \{q,q'\}} \Pr[L(q') = q_{HL}] + \sum_{q_{HL} \in Y \cap q,q'} \Pr[L(q) = q_{HL}] \qquad (7.2) \\
&\leq \sum_{q_{HL} \in Y \setminus \{q,q'\}} \Pr[L(q') = q_{HL}] + \sum_{q_{HL} \in Y \cap \{q,q'\}} \big(e^{\epsilon'_Q} \Pr[L(q') = q_{HL}] + \frac{\delta'_Q}{2}\big) \qquad (7.3) \\
&\leq e^{\epsilon'_Q} \sum_{q_{HL} \in Y} \Pr[L(q') = q_{HL}] + 2 \cdot \frac{\delta'_Q}{2} \\
&= e^{\epsilon'_Q} \Pr[L(q') \in Y] + \delta'_Q,
\end{aligned}$$

Equality (7.2) stems from the fact that the probability of reporting a false query is independent of the user's true query. The inequality (7.3) is a direct application of inequality (7.1). 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 [17], each of the at most m_C iterations of L is $(\epsilon'_Q + \epsilon'_U, \delta'_Q + \delta'_U) = (\epsilon', \delta')$-differentially private. □

Observation 3.4 . *$\hat{p}_C$ 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,

$$\begin{aligned}
r_{C,q} &= t \cdot p_q + \sum_{q' \neq q} p_{q'}(1-t)\frac{1}{k-1} \\
&= t \cdot p_q + \frac{1-t}{k-1} \sum_{q' \neq q} p_{q'} \\
&= t \cdot p_q + \frac{1-t}{k-1}(1-p_q).
\end{aligned}$$

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 dataset. Then $r_{C,\langle q,u\rangle} = t \cdot t_q \cdot p_{\langle q,u\rangle} + \left(t\frac{1-t_q}{k_q - 1}\right)(p_q - p_{\langle q,u\rangle}) + \left(\frac{1-t}{k-1} \cdot \frac{1}{k_q}\right)(1 - p_q)$, 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 $\hat{p}_{C,\langle q,u\rangle} = \frac{\hat{r}_{C,\langle q,u\rangle} - \left(t\frac{1-t_q}{k_q-1}\hat{p}_{C,q} + \frac{(1-t)(1-\hat{p}_{C,q})}{(k-1)k_q}\right)}{t(t_q - \frac{1-t_q}{k_q-1})}$. □

Theorem 3.5 . *If $m_O = 1$ then the unbiased variance estimate for the opt-in group's record probabilities 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 $Y \sim \text{Laplace}(b_T)$ with b_T being the scale parameter and recalling that $|D_T|$ 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 $X_j \sim \text{Bernoulli}(p_{\langle q,u\rangle})$ is the random variable indicating whether report j was record $\langle q, u\rangle$.

The expectation of this estimator is given by $\mathrm{E}[\hat{r}_{O,\langle q,u\rangle}] = p_{\langle q,u\rangle}$. 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

$$\begin{aligned}
\sigma^2_{O,\langle q,u\rangle} &= \mathrm{Var}[\hat{p}_{O,\langle q,u\rangle}] && (7.4)\\
&= \mathrm{Var}\Big[\frac{1}{|D_T|}\Big(\sum_{j=1}^{|D_T|} X_j + Y\Big)\Big] \\
&= \frac{1}{|D_T|^2}\Big(\mathrm{Var}\Big[\sum_{j=1}^{|D_T|} X_j\Big] + \mathrm{Var}\,[Y]\Big) && (7.5)\\
&= \frac{1}{|D_T|^2}\Big(\sum_{j=1}^{|D_T|} \mathrm{Var}\,[X_j] + \mathrm{Var}\,[Y]\Big) && (7.6)\\
&= \frac{1}{|D_T|^2}\big(|D_T| \cdot p_{\langle q,u\rangle}(1 - p_{\langle q,u\rangle})\big) + 2\Big(\frac{b_T}{|D_T|}\Big)^2 \\
&= \frac{p_{\langle q,u\rangle}(1 - p_{\langle q,u\rangle})}{|D_T|} + 2\Big(\frac{b_T}{|D_T|}\Big)^2.
\end{aligned}$$

Equality 7.5 comes from the independence between Y and all X_j. Equality 7.6 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).

To 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[11]") to generate an unbiased estimate. Thus, the variance of each opt-in record probability estimate is: $\hat{\sigma}^2_{O,\langle q,u\rangle} = \frac{|D_T|}{|D_T|-1}\left(\frac{\hat{p}_{O,\langle q,u\rangle}(1-\hat{p}_{O,\langle q,u\rangle})}{|D_T|} + 2\left(\frac{b_T}{|D_T|}\right)^2\right)$. □

Theorem 3.6 . *If $m_C = 1$ then the unbiased variance estimate for the client group's record probabilities can be computed as:*

$$\hat{\sigma}^2_{C,\langle q,u\rangle} = \frac{|D_C|}{t^2\left(t_q - \frac{1-t_q}{k_q-1}\right)^2(|D_C|-1)}\cdot$$
$$\Big(\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,\langle q,u\rangle})}{|D_C|} + \big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)^2\hat{\sigma}^2_{C,q} + 2\big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)\frac{\hat{r}_{C,\langle q,u\rangle}(1-\hat{r}_{C,q})}{|D_C|(t-\frac{1-t}{k-1})}\Big).$$

Proof. We'll first derive the variance estimate for the client group's query probabilities, then move on to the variance estimate for their record probabilities.

From the proof of Observation 3.4, the distribution of the reported query q from the client algorithm 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 recalling that $|D_C|$ is the total number of records from the client users.

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

$$\begin{aligned}
\text{Var}[\hat{r}_{C,q}] &= \text{Var}\Big[\frac{1}{|D_C|}\sum_{j=1}^{|D_C|} X_j\Big] \\
&= \Big(\frac{1}{|D_C|}\Big)^2\sum_{j=1}^{|D_C|}\text{Var}\left[X_j\right] &(7.7)\\
&= \big(\frac{1}{|D_C|}\big)^2\big(|D_C|\cdot r_{C,q}(1-r_{C,q})\big) &(7.8)\\
&= \frac{r_{C,q}(1-r_{C,q})}{|D_C|},
\end{aligned}$$

where equality 7.7 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).

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

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

To 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

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

algorithm is: $\hat{\sigma}^2_{C,q} = \left(\frac{1}{t - \frac{1-t}{k-1}}\right)^2 \frac{\hat{r}_{C,q}(1-\hat{r}_{C,q})}{|D_C|-1}$.

Now, we'll derive the variance estimate for the record probabilities. For a given query q and corresponding URL u in head list, denote X_i^q as the indicator random variable that is 1 if user i reported query q and 0 otherwise, and similarly denote $X_i^{\langle q,u\rangle}$ as the indicator random variable that is 1 if user i reported query q and URL u and 0 otherwise. Note that $X_i^q \sim \text{Bern}(r_{C,q})$ and $X_i^{\langle q,u\rangle} \sim \text{Bern}(r_{C,\langle q,u\rangle})$. The covariance between these two random variables is given by

$$\text{Cov}[X_i^q, X_i^{\langle q,u\rangle}] = \text{E}[X_i^q X_i^{\langle q,u\rangle}] - \text{E}[X_i^q]\text{E}[X_i^{\langle q,u\rangle}] = r_{C,\langle q,u\rangle} - r_{C,\langle q,u\rangle} r_{C,q} = r_{C,\langle q,u\rangle}(1 - r_{C,q}).$$

Also note that due to the iid assumption, for any other user j, we have $\text{Cov}(X_i^q, X_j^{\langle q,u\rangle}) = 0$. Thus, we have the covariance between our empirical query and record estimates as

$$\begin{aligned}
\text{Cov}[\hat{r}_q, \hat{r}_{\langle q,u\rangle}] &= \text{Cov}\left[\frac{1}{|D_C|}\sum_{i\in D_C} X_i^q, \frac{1}{|D_C|}\sum_{i\in D_C} X_i^{\langle q,u\rangle}\right] \\
&= \frac{1}{|D_C|^2}\text{Cov}\left[\sum_{i\in D_C} X_i^q, \sum_{i\in D_C} X_i^{\langle q,u\rangle}\right] \\
&= \frac{1}{|D_C|^2}\sum_{i,j\in D_C} \text{Cov}[X_i^q, X_j^{\langle q,u\rangle}] \\
&= \frac{1}{|D_C|^2}\sum_{i\in D_C} \text{Cov}[X_i^q, X_i^{\langle q,u\rangle}] \\
&= \frac{r_{C,\langle q,u\rangle}(1 - r_{C,q})}{|D_C|}.
\end{aligned}$$

Utilizing this covariance expression, we can now compute the desired variance estimate as:

$$
\begin{aligned}
&\sigma^2_{C,\langle q,u\rangle} = \mathrm{Var}[\hat{p}_{C,\langle q,u\rangle}] \\
&= \mathrm{Var}\left[\frac{\hat{r}_{C,\langle q,u\rangle} - \big(t\frac{1-t_q}{k_q-1}\hat{p}_{C,q} + \frac{(1-t)(1-\hat{p}_{C,q})}{(k-1)k_q}\big)}{t(t_q - \frac{1-t_q}{k_q-1})}\right] \\
&= \frac{1}{t^2(t_q - \frac{1-t_q}{k_q-1})^2}\mathrm{Var}\left[\hat{r}_{C,\langle q,u\rangle} - \big(t\frac{1-t_q}{k_q-1}\hat{p}_{C,q} + \frac{(1-t)(1-\hat{p}_{C,q})}{(k-1)k_q}\big)\right] \\
&= \frac{1}{t^2(t_q - \frac{1-t_q}{k_q-1})^2}\mathrm{Var}\left[\hat{r}_{C,\langle q,u\rangle} - \hat{p}_{C,q}\big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)\right] \\
&= \frac{1}{t^2(t_q - \frac{1-t_q}{k_q-1})^2}\cdot \\
&\quad\left(\mathrm{Var}\left[\hat{r}_{C,\langle q,u\rangle}\right] + \big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)^2\mathrm{Var}\left[\hat{p}_{C,q}\right] + 2\big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)\mathrm{Cov}[\hat{p}_{C,q}, \hat{r}_{C,\langle q,u\rangle}]\right) \\
&= \frac{1}{t^2(t_q - \frac{1-t_q}{k_q-1})^2}\cdot \\
&\quad\left(\frac{r_{C,\langle q,u\rangle}(1-r_{C,\langle q,u\rangle})}{|D_C|} + \big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)^2\sigma^2_{C,q} + 2\big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)\frac{1}{t-\frac{1-t}{k-1}}\mathrm{Cov}[\hat{r}_{C,q}, \hat{r}_{C,\langle q,u\rangle}]\right) \\
&= \frac{1}{t^2(t_q - \frac{1-t_q}{k_q-1})^2}\cdot \\
&\quad\left(\frac{r_{C,\langle q,u\rangle}(1-r_{C,\langle q,u\rangle})}{|D_C|} + \big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)^2\sigma^2_{C,q} + 2\big(\frac{1-t}{(k-1)k_q} - t\frac{1-t_q}{k_q-1}\big)\frac{1}{t-\frac{1-t}{k-1}}\frac{r_{C,\langle q,u\rangle}(1-r_{C,q})}{|D_C|}\right).
\end{aligned}
$$

Using our already-computed estimates $\hat{r}_{C,q}, \hat{r}_{C,\langle q,u\rangle}$, and $\hat{\sigma}^2_{C,\langle q,u\rangle}$ (in place of $r_{C,q}, r_{C,\langle q,u\rangle}$, and $\sigma^2_{C,\langle q,u\rangle}$ respectively) and applying Bessel's correction, we obtain the stated result. □

Theorem 3.7 . *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, and the blended estimate is the convex combination $\hat{p}_{\langle q,u\rangle} = w_{\langle q,u\rangle}\cdot\hat{p}_{O,\langle q,u\rangle} + (1-w_{\langle q,u\rangle})\cdot\hat{p}_{C,\langle q,u\rangle}$, then the sample variance optimal weighting is given by $w_{\langle q,u\rangle} = \frac{\hat{\sigma}^2_{C,\langle q,u\rangle}}{\hat{\sigma}^2_{O,\langle q,u\rangle}+\hat{\sigma}^2_{C,\langle q,u\rangle}}$.*

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